VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Sémantický web a jeho využití

Semantic Web and Its Utilization

2019 Jiří Lajčok

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Lajčok**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Sémantický web a jeho využití
Semantic Web and Its Utilization

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je zmapovat možnosti moderní integrace sémantických informací do webových stránek a aplikací, ilustrovat možnosti a typické způsoby využití.

1. Zmapujte a popište současné aktuální možnosti integrace sémantických dat do webových stránek.
2. Navrhněte aplikaci, která bude schopna získávat sémantické informace různého typu z webových stránek. Zároveň bude umožňovat ověřit správnost integrace.
3. Implementujte navržený nástroj s využitím webových technologií (HTML5).
4. Na případových studiích ilustrujte možnosti vkládání sémantických dat a funkčnost aplikace.
5. Zhodnoťte aktuální stav a trendy sémantických dat ve webových stránkách.

Seznam doporučené odborné literatury:

[1] Jon Duckett: JavaScript and JQuery: Interactive Front-End Web Development, Wiley, 2014, ISBN: 978-1118531648
[2] Erixc Elliot: Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries, O'Reilly Media, 2014, ISBN: 978-1491950296
[3] Leslie Sikos: Mastering Structured Data on the Semantic Web: From HTML5 Microdata to Linked Open Data, Apress, 2015, ISBN: 978-1484210505

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2019 .....

Rád bych poděkoval vedoucímu mé práce Ing. Michalu Radeckému, Ph.D. za ochotu při konzultacích a za cenné, praktické rady. Dále děkuji své rodině a svým blízkým za podporu jak při studiu, tak i v životě.

Abstrakt

Práce se zabývá fenoménem sémantického webu a technikám integrace jeho základního stavebního kamene – strukturovaných dat. Většina obsahu na internetu je určena pro lidi, ale s rostoucím množstvím informací je zapotřebí zefektivnit práci s nimi. Vhodným podáním dat lze strojům umožnit hlubší pochopení jejich významu a získávat tak relevantnější výsledky nejen při vyhledávání.

Součástí je také aplikace demonstrující získávání dat z webových dokumentů. Aplikace a údaje z ní obdržené slouží jako podklady pro případové studie na reálných subjektech jako jsou například e-commerce weby.

Klíčová slova: sémantický web, strukturovaná data, webové technologie

Abstract

The thesis is dealing with the semantic web phenomenon and the integration techniques of the structured data – the foundations of the semantic web. Most of the internet content is made for the people but with the amount of informations growing, it is essential to manage them more effectively. We are able to get the machines to understand the meaning of the data by serving it to it the right way and thus get more relevant results not only with searching.

There is also an application as a part of the work to demonstrate extraction of the data from web documents. The results obtained from the application are being used as data sets for the case studies on actual subjects such as e-commerce sites.

Key Words: semantic web, structured data, web technologies

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Public Interface
CSS	–	Cascading Style Sheets
DOM	–	Document Object Model
HTML	–	Hyper Text Markup Language
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Secure
IP	–	Internet Protocol
IRI	–	Internationalized Resource Identifier
ISBN	–	International Standard Book Number
JSON	–	JavaScript Object Notation
JSON-LD	–	JavaScript Object Notation for Linked Data
JSX	–	JavaScript XML
NFC	–	Near Field Communication
NPM	–	Node Package Manager
QR	–	Quick Response
RDF	–	Resource Description Framework
RDFa	–	Resource Description Framework in attributes
SPA	–	Single Page App
URI	–	Uniform Resource Identifier
URL	–	Uniform Resource Locator
XML	–	Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Seznámení s tématem

Na webu se nachází obrovské množství informací různorodého typu, formátu či kvality. Jedná se o metaforickou kupku sena a zdá se až nepravděpodobné, že v ní najdeme právě to, co hledáme. Při takovém objemu rostou nároky na organizaci dat spojenou s jejich uchováváním a zpracováním za cílem maximální efektivity práce s nimi. K orientaci v tomto informačním přebytku nám pomáhají webové katalogy, vyhledávače či porovnávače.

Zatímco se vyhledávače staly jedním z primárních vstupních bodů do internetu a míra jejich přesnosti je v jistých případech oslnivá, vzniká snaha o ještě hlubší pochopení obsahu vyhledávacími algoritmy – pomocí *strukturovaných dat*. Díky nim je pro stroj možné detekovat význam (*sémantiku*) obsahu [1].

Rozdíl v chápání stroje volně psaného textu a strukturovaných dat lze přiblížit na příkladu z knihy L. Sikose: Pokud vyhledávači zadáme pojem `jaguar`, robot nerozezná rozdíl mezi kočkovitou šelmou a výrobcem automobilů [2]. Algoritmus totiž pracuje převážně s výskyty hledaného slova v textu, případně v názvu obrázku, videa, atp. Existují pokusy využívající strojového učení např. k detekci objektů v obrázku, ale takový přístup je zatím neefektivní a nespolehlivý. Tento problém lze vyřešit přiložením strukturovaných dat s dodatečnými sémantickými informacemi a kontextem.

Strukturovaná data kromě toho umožňují také přehlednější formu prezentace informací koncovému uživateli. Příkladem může být přehledná tabulka odletů místo úryvků textů ze stránek letecké společnosti. Za zmínku stojí také komunikace s hlasovými asistenty formou otázek a smysluplně formulovaných odpovědí. Jedná se rovněž o alternativní formu poskytování obsahu na webu, kterou lze vhodným zpracováním poskytovat i lidem s postižením.

Sémantický web je jedním z klíčových milníků pro tzv. Web 3.0, ve kterém oproti předchozí verzi 2.0 dochází k přenesení důrazu na data (informace) spíše než na dokumenty [2].

2 Definice pojmů

Pro bližší uchopení problematiky je nutné ustálit několik pojmů jako jsou *sémantika* či *strukturovaná data*. Informace v této sekci ve velké míře těží ze zdrojů [2, 3], další budou vyjádřeny explicitně.

2.1 Sémantika

Jedná se o význam kódu či jiné konstrukce ve skutečnosti, jinými slovy kód reprezentuje jakýsi reálný objekt, činnost. I věta v lidském jazyce je pouze kombinací písmen, která však dohromady, při interpretaci (porozumění), reprezentují informaci. Pro snadnější orientaci je kód typicky dělen do bloků, funkcí nebo jiných struktur. Jakýkoliv způsob třídění s sebou nese společný element: vznikají paralely ke skutečnosti, jak ji vnímá člověk.

V této práci je pojem *sémantika* konkretizován na samotný smysl informací na webu obsažených. Neměl by být zaměňován například se *sémantickými* značkami v HTML5. Ačkoliv spolu témata částečně souvisí, elementy v HTML5 jsou určeny spíše k vyznačení významu struktury dokumentu (rozlišení navigace od hlavního obsahu). Zároveň je vývoj specifikace HTML příliš pomalý, než aby stíhal rychlosti vývoje informací na webu [1].

2.2 Strukturovaná data

Aby stroj dokázal interpretovat (pochopit) informaci, vyžaduje to použití *formálního jazyka*[1], kterým lidská řeč není. K tomuto účelu slouží *strukturovaná data*, která jsou formálním jazykem, zpravidla se snadno čtou a lze je jednoznačně interpretovat. Obecně můžeme jako strukturovaná data označit například QR kódy, díry v děrném pásku nebo kódování informací přenesených dotykem zařízení NFC (nejedná se o vizuální reprezentaci dat, ale princip zůstává stejný). V internetovém prostředí jsou pak data běžně strukturována například do formátů JSON či XML.

2.3 Propojená data

Propojená data (anglicky linked data) jsou strukturovaná data, která obsahují reference na jiné, externí dokumenty (rovněž strojově zpracovatelné). Fungují podobně jako hypertextové odkazy na běžném webu. Dobrým příklad může být Wikipedie, která odkazování hojně využívá. Věnuje-li se článek nějaké psí rase, jistě se v něm vyskytne mnohokrát slovo `pes`. Avšak místo obecného popisu tohoto tvora bude slovo `pes` označeno jako odkaz směřující na samostatný článek o něm.

Velmi podobně fungují propojená data – pokud existuje zdroj, který již obsahuje nějaká data spojená s tématem, využije se raději odkazu než vytváření duplicit. To má za následek diverzitu zdrojů, z nichž se každý může věnovat konkrétní problematice dopodrobna spíš než širokému tématu povrchně.

[1] Jazyk, který se řídí pevně definovanými pravidly, gramatikou

Je také žádoucí, aby pro každou informaci, na kterou lze smysluplně odkázat, byla reference uvedena. Čím je provázanost dokumentů hustější, tím je větší šance, že s jejich pomocí stroj nalezne cestu k požadovanému obsahu.

2.4 Sémantický web

Většina obsahu na webu je určena ke konzumaci lidskými bytostmi. Známe již potencionální přínosy strojově čitelného obsahu. Sémantický web je ve své podstatě web pro stroje, určený k tomu, aby na něm roboti prováděli svou práci, procházeli jím a pohybovali se z jednoho webu na druhý díky propojům – referencím. Výsledkem je globální databáze ve strukturovaném formátu, oproti běžnému webu hustěji propojovaná a lépe organizovaná.

S filozofií sémantického webu se pojí pojmy jako *znovupoužitelnost*, *provázanost* a *decentralizovanost*. V ideálním případě nikdy neobsahuje informaci, kterou je možné získat z jiného libovolného zdroje. Jsou v něm popsány pouze originální informace, ostatní je možné uvést pomocí referencí. Je-li tedy popisován například pes, definice tohoto zvířete – „psa“ – je realizována odkazem na jiný vhodný zdroj. V dokumentu se dále nachází pouze definice toho jednoho konkrétního.

Díky využití propojených dat v sémantickém webu lze získávat data z této informační báze pomocí *sémantický dotazů*, které díky propojům mají přístup k informacím, které by nebylo možné najít v původním dokumentu [2].

Ačkoliv je žádoucí recyklovat funkci existujících entit, stále je možné vytvářet vlastní obsah a volně je publikovat. Pro identifikaci objektů a dokumentů na sémantickém webu slouží URL, neexistuje tedy žádná centrální autorita, která by sémantický web měla ve správě. Sémantický web fakticky představuje podmnožinu obecného webu, vztahují se k němu tedy naprosto stejné podmínky, ať už pravidla či práva.

2.5 Ontologie

Ontologie coby věda o bytí a elementárních pojmech světa je zároveň označením deskriptivního nástroje k tvorbě sémantických struktur. Jejími elementárními částmi jsou *pojmy*[2] a vztahy[3] mezi nimi. Pochopitelně jsou definice poskytovány opět formou strukturovaných dat.

Základní ontologie se skládají z následujících komponent:

- *Třídy* – Abstrakce typů objektů, jejich skupin nebo kolekcí, anglicky *class*. Setřiďují objekty sdílející společné *vlastnosti*. Umožňují dědičnost (i vícenásobnou), která je užitečná k dědění vlastností potomkem nebo pouze za účelem organizování tříd do jisté hierarchie[4].
- *Vlastnosti* – Zpravidla dvojice *název–hodnota*, anglicky *property* nebo *attribute*. Náleží vždy do nějaké třídy objektů.

[2] Výraz pro popisovanou entitu v ontologiích, anglicky obvykle *term*
[3] Rekurzivně se jedná rovněž o pojem, ontologie je *popisuje* jako vztah
[4] Nemusí se vždy jednat o stromovou strukturu, zejména v případě mnohonásobného dědění

- *Instance* – Objekt nebo výskyt určité třídy. Slouží například pro odlišení seznamu zaměstnanců na oddělení (ve smyslu osob) od seznamu typů (tříd s názvem `Zaměstnanec`).
- *Relace* – Logická vazba mezi třídami, instancemi, mezi instancí a třídou, mezi jednou instancí a kolekcí nebo mezi kolekcemi navzájem.
- *Omezení* – Formální definice typů a rozsahů hodnot vlastností.

Strukturu ontologií lze reprezentovat grafem (ať už ve smyslu hierarchie dědičnosti tříd nebo relací mezi třídami), platí v něm tranzitivita[5], v ontologiích tato vlastnost umožňuje tzv. *odvození*. Platí-li například tvrzení „*Petr žije v Ostravě*“ a zároveň „*Ostrava je českým městem*“, pak je také *odvozené* tvrzení „*Petr žije v Česku*“ platné.

2.6 Slovníky

Je-li ontologie nástroj pro definici sémantiky, pak je sada definic označována jako slovník nebo také schéma. Je žádoucí, aby ontologie umožňovaly rozšiřování slovníků. Snižuje se tak pravděpodobnost vytváření duplicit popisujících jednu a tu samou věc. Platí, že se raději použije existující a rozšíří se o vlastní pojmy. Slovníky se zpravidla zaměřují na určitou problematiku, neboli doménu.

Slovník se identifikuje jednoznačnou absolutní adresou URL[6], která tvoří základ jmenného prostoru obsažených pojmů. Pojem lze ve slovníku identifikovat různými způsoby:

- Slovník je umístěn přímo v kořenu doménového jména, absolutní adresa pojmu pak může vypadat takto:
  `http://example.com/HledanyPojem`
- Záleží na celém doménovém jméně, proto následující příklad identifikuje jiný pojem v jiném slovníku:
  `http://slovnik.example.com/HledanyPojem`
- Jmenný prostor lze upřesnit také pomocí specifikací cesty:
  `http://example.com/cesta/ke/slovniku/HledanyPojem`
- Celý slovník se může také nacházet v jediném dokumentu, pojem pak specifikujeme jako jeho fragment:
  `http://example.com/slovnik.rdf#HledanyPojem`

V rámci slovníku lze pro zjednodušení obvykle odkazovat na jiné pojmy relativně uvedením pouze názvu pojmu. Pro odkaz na pojem z jiného slovníku se zpravidla užívá absolutní adresy. Některé ontologické formáty podporují vytváření aliasů pro zkrácení zápisu.

[5] Formálně matematicky zapsáno: $\forall \alpha, \beta, \gamma \in X, \alpha R \beta \wedge \beta R \gamma \Rightarrow \alpha R \gamma$, kde X je množina vrcholů

[6] Přípustné je i použití adresy IP, ale nejedná se o běžnou praxi [2]

`Schema.org` Jde o nejkompletnější kolekci slovníků, kterou v roce 2011 založili Google, Yahoo! a Bing. Skládá se z jádra (slovník se základními pojmy), ke kterému jsou napojeny další, rozšiřující slovníkové moduly. Jádro slouží k popisu těch nejběžnějších věcí a scénářů (například událostí, organizací a osob). Rozšiřující slovníky se pak věnují specifičtějším doménám, mezi oficiální patří například `bib.schema.org` přidávající pojmy pro bibliografii. Schema.org lze dále volně rozšiřovat, vznikají tak neoficiální komunitní slovníky, které je možno publikovat samostatně nebo přímo na `schema.org`. Výsledkem je neustále rostoucí zásoba pojmů podporovaná a rozšiřovaná nejen velkými společnostmi, ale také komunitou [2, 4]. Od této části náleží všechny pojmy v práci do jmenného prostoru `schema.org`.

3 Praktická využití

Ačkoliv teoretický potenciál sémantického webu byl nastíněn již v úvodní sekci č. 1, tato část práce mapuje pragmatické faktory motivující k integraci strukturovaných dat. Ať už za cílem zvýšení dosahu a relevance ve vyhledávačích, zařazení do porovnávačů nebo pro budoucí využití.

3.1 Konzumenti

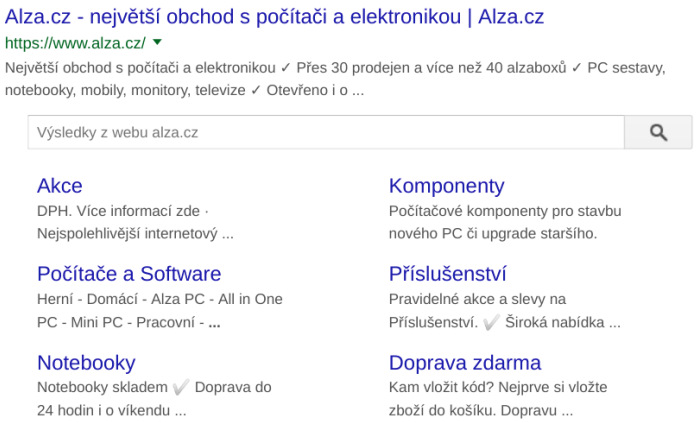
Existují dvě hlavní skupiny konzumentů obsahu na internetu: webové prohlížeče a vyhledávače [1]. Prohlížeč je zpravidla obsluhován člověkem, zatímco indexování obsahu na webu provádí vyhledávací roboti (anglicky se používá též pojem *crawler*).

Strukturovaná sémantická data mohou robotu pomoci přesněji interpretovat obsah stránky a tím ji správně kategorizovat (viz příklad s výrazem `jaguar` v sekci 1) [2]. Kromě toho mohou obdržené informace také efektivněji prezentovat koncovému uživateli [1].

3.2 Vyhledávače

Globálně nejpoužívanější vyhledávač Google [5] poskytuje manuál k užití strukturovaných dat za účelem *obohacených výsledků* (v anglickém originále označováno jako *rich results*). Obohacené výsledky nabízí vedle klasického zobrazení (titulek stránky, adresa URL a výňatek či popisek webu) také rozšiřující prvky jako vyhledávací panel či výpis z navigace [6].

Různé typy položek se rozdílně i zobrazují – od tzv. *breadcrumb* navigace (podrobná navigace zanoření stránky) zobrazené na místě adresy až po karty představující jednotlivé články, recepty atp. Cílem je zaujmout uživatele a zobrazit mu přehledné, relevantní informace. Provozovatelé webu zase pozorují zvýšenou návštěvnost [7].



Obrázek 1: Pohled na obohacené výsledky vyhledávání

Robot Google zpracovává strukturovaná data také pro databázi *Google Knowledge Graph*. Jedná se o znalostní bázi osob, organizací, míst a obecně věcí. Pokud je vyhledávaný termín

identifikován v bázi, zobrazí se výsledek po boku běžných výsledků, případně nad nimi. Mimo to lze bázi využít i prostřednictvím API pro registrované [8].

Obrázek 2: Dotazování na znalostní bázi společnosti Google

3.3 Virtuální asistenti

Virtuální asistenti, často zároveň hlasoví, nabízejí moderní způsob získávání informací kladením otázek jako člověku. Po interpretaci otázky v lidské řeči umělou inteligencí začnou asistenti vyhledávat relevantní informace. Činí tak preferovaně ve znalostních bázích (vyhledávání zpravidla slouží jako záložní řešení) a získané výsledky zpětně formulují do lidské řeči jako odpověď. Znalostní báze (viz dříve zmíněná *Google Knowledge Graph* využívaná asistentem Google) jsou tvořeny právě sbíranými strukturovanými daty na sémantickém webu [8].

4 Technická řešení

RDF (Resource Description Framework – framework pro popis zdrojů) je standardním nástrojem pro specifikaci strukturovaných i sémantických dat. Jedná se však spíš o přístup k problematice než o konkrétní formát. Dle principu RDF se řídí několik formátů, z nichž některé jsou vhodné pro použití na webu [2].

4.1 Datový model

Datový model strukturovaných dat je zpravidla shodný nehledě na použitou technologii (s výjimkou případných omezení, která budou zmíněna). Model obsahuje skupiny párů *název–hodnota*. Samotný pár se běžně nazývá *vlastnost* (anglicky *property*) a skupina vlastností je označována jako *položka* (anglicky *item*). Každá položka může mít definovaný typ (či více typů) a globální identifikátor (ve vztahu jako ISBN ku knize). Hodnotami vlastností mohou být buď text, nebo vnořená položka [9, 10].

Terminologie částečně souvisí s pojmy probranými v sekci o ontologiích (sekce č. 2.5), avšak nyní jsou definována spíše z pohledu samotných dat než slovníkové tvorby.

Položka	**Typ**	**Vlastnost**	**Hodnota**
`Item[1]`	`Person`	`name`	`Jiri Lajcok`
		`birthDate`	`1997`
		`affiliation`	`Item[2]`
`Item[2]`	`CollegeOrUniversity`	`url`	`https://www.vsb.cz/`
		`name`	`VSB`

Tabulka 1: Příklad tabulkové reprezentace strukturovaných dat

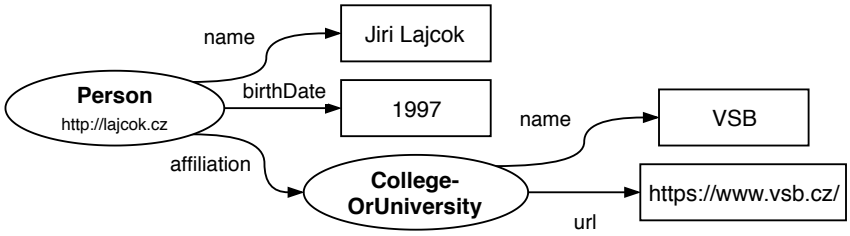
Slovník `schema.org` specifikuje také datové typy. Mezi primitivní patří například `Text`, `URL` či `Number` a jsou zastřešeny pojmem `DataType`. Hodnota položky je zpravidla vždy pouhý řetězec, ale datové typy (primitivní) plní funkci restrikce. K dispozici jsou také složené datové typy jako `Enumeration` – enumerace. `Schema.org` přistupuje k problematice pragmaticky, s čímž souvisí některé rysy tohoto slovníku jako mnohonásobná dědičnost nebo fakt, že datové typy `Text` a `URL` mohou být přiřazeny kterékoliv vlastnosti. To vše dle filozofie „raději nějaká informace než-li žádná“ [4].

4.2 Grafická reprezentace

Strukturovaná data lze reprezentovat grafem – konkrétně libovolným počtem nezávislých kořenových stromů[7], označovaným také jako les. Vrcholy grafu představují hodnoty (položky či přímo hodnoty vlastností), hrany pak jednotlivé vlastnosti (zpravidla obsahující název jako popisek).

[7]Strom ve smyslu jako souvislý, acyklický graf; orientovaný strom (nebo-li kořenový) navíc obsahuje kořen a lze u něj rozlišovat úrovně vzhledem ke kořenu

Hrany směřují vždy a pouze z položek (nikdy z hodnot). Kořeny stromů odpovídají nejvyšším položkám v dokumentu. Tato reprezentace je pochopitelně rovněž nezávislá na použité technologii [9, 10].



Obrázek 3: Grafická reprezentace strukturovaných dat

Položka typu `Person` na obr. 3 představuje kořen stromu (je položkou nejvyšší úrovně). Hrany jsou popsány (`name`, `birthDate`) pro specifikaci názvů vlastností a směřují do hodnot (`1997`) nebo podřadných položek (`CollegeOrUniversity`). Položka může způsobovat další rozvětvení.

4.3 Integrace strukturovaných dat

Již dříve před implementací Microdat do HTML5 či spuštění `schema.org` v roce 2015 bylo na webu možné specifikovat strukturovaná data. Oblíbený formát byl zejména Microformats recyklující funkce atributů HTML (konkrétně `rel`, `rev` a `class`) pro definici strojově čitelných dat v HTML. Časem byl však kvůli svým omezením a problémům vytlačen jinými řešeními [1, 2].

Početné skupině technologií stojí jako předloha RDF. Patří mezi ně například RDFa (určené pro XHTML, následně implementováno i do HTML), JSON-LD. RDFa využívá podobně jako Microdata atributů značek HTML, kdežto JSON-LD je kódován notací JSON. Všechny tyto formáty jsou sice zapisovány jiným způsobem, ale zachovávají stejné možnosti [2].

4.3.1 HTML5 Microdata

Standard HTML5 se zaměřil na sémantiku v širším slova smyslu – ve smyslu logického členění kódu a obsahu raději než vizuálního a také ve smyslu zadávání strukturovaných dat. Předmětem prvního jsou nové sémantické značky – `<header>`, `<nav>`, `<section>` atd. Ač jsou pro základní strojové procházení webu užitečné, v našem případě se zaměříme na případ druhý – sémantiku dle technologie *Microdata* [1].

Sémantické značky jsou dány a do další verze standardu se jen tak nezmění ani nepřibudou. Takové tempo však nestačí rychlosti vzniku a rozšiřování informací. Microdata na druhou stranu představují nástroj pro hlubší sémantické označkování pojmy z externích slovníků. Nepřidávají do HTML žádnou novou značku, místo toho se využívá atributů, které mohou být přiřazeny takřka jakémukoliv elementu [9]:

- `itemscope` – Hraniční značky elementu označeném tímto atributem ohraničují prostor pro popisné vlastnosti dané položky. Není očekávaná žádná hodnota.

- `itemref` – Slouží pro rozšíření kolekce elementů zkoumaných na vlastnosti. Identifikátory (hodnota atributu, oddělené mezerami) specifikují elementy s odpovídajícím `id`, které se přidají do kolekce (včetně potomků).
- `itemtype` – Definuje třídu (či více tříd) položky ve formě mezerami dělených URL adres, které odkazují na slovníkové pojmy. Nesmí se vyskytovat jinde než na elementu s atributem `itemscope`.
- `itemid` – Hodnotou lze specifikovat identifikátor položky na elementu s `itemscope`. Obsahuje například ISBN v případě popisu knihy.
- `itemprop` – Označuje element představující vlastnost položky. Hodnotou je název vlastnosti – relativní (vzhledem k jmennému prostoru typu položky) či absolutní (nahrazuje jmenný prostor typu jiným). Pro specifikaci více vlastností najednou je lze zadat oddělené mezerami jako hodnotu atributu.

  Samotná hodnota vlastnosti je vyhodnocena pro daný element jednou z následujících strategií:

  1. Pokud je element zároveň označen atributem `itemscope`, bude hodnota interpretována jako položka (nastává rekurzivní vyhodnocení podřazené položky).
  2. Existují elementy, pro které je uplatňována speciální strategie. Například pro `<a>`, `<link>` a jiné odkazovací konstrukce bude použita hodnota atributu `href`, u `<img>` pak atribut `src` a pro `<time>` je to `datetime` (pokud je specifikován). Na všech hodnotách atributů sloužících jako odkazy navíc dochází k vyhodnocení absolutní URL adresy v závislosti na aktuální zdrojové adrese.
  3. Pro všechny ostatní elementy je použit textový obsah mezi značkami, konkrétně hodnota `textContent` v DOM.

Následující vstupní kód popisuje osobu bez vyznačení Microdaty:

```
<section>
   <h1>Predstaveni</h1>
   <p>
      Dobry den,
      jmenuji se Jiri Lajcok,
      je mi <time datetime="1997">22 let</time>
      a studuji na <a href="https://www.vsb.cz/">VSB</a>.
   </p>
</section>
```

Výpis 1: Vstupní kód HTML před označením

Stroj prochází kódem 4.3.1 a narazí na počáteční tag sekce `<section>`, následuje její povinný nadpis `<h1>`. Dále prochází odstavcem `<p>`, jehož obsah pro něj nemá žádný význam. Nakonec narazí na značku času `<time>`, podle atributu `datetime` (obsahuje časový údaj ve standardizovaném formátu) pozná, že čas vyjádřený textem uvnitř elementu znamená rok 1997. Díky HTML5 značkám jsme si tedy trochu pomohli v případě sekce s nadpisem (použije se při konstrukci stromové struktury nadpisů v dokumentu) a strojově čitelným časovým údajem. Nakonec registruje odkaz směřující na adresu v atributu `href`.

Následuje zdroj označený Microdaty:

```
<section
      itemscope
      itemtype="http://schema.org/Person"
      itemid="http://lajcok.cz"
>
   <h1>Predstaveni</h1>
   <p>
      Dobry den,
      jmenuji se <span itemprop="name">Jiri Lajcok</span>,
      je mi <time datetime="1997" itemprop="birthDate">22 let</time>
      a studuji na
      <span
            itemscope
            itemtype="http://schema.org/CollegeOrUniversity"
            itemprop="affiliation"
      >
         <a href="https://www.vsb.cz/" itemprop="url">
            <span itemprop="name">VSB</span>
         </a>
      </span>.
   </p>
</section>
```

Výpis 2: Kód HTML po označení Microdaty

Algoritmus při návštěvě elementu `<section>` registruje atribut `itemscope` (a zároveň absenci `itemprop`) a vytváří novou položku. Typ položky definuje podle `itemtype` jako osobu `Person` ze slovníku `schema.org`. Položce je dále přiřazen identifikátor z atributu `itemid`. Algoritmus přechází k mapování vlastností položky a mezi potomky elementu nachází elementy s `itemprop`. Jako první narazí na vlastnost `name`, které přidá hodnotu z textu mezi značkami. Nakonec detekuje `itemprop="birthDate"` a podle specifického chování pro tag `<time>` přiřazuje hodnotu z atributu `datetime`. Na následné složené konstrukce nejprve nalezne vlastnost

`itemprop="affiliation"` a očekává hodnotu. Atribut `itemscope` mu však značí počátek nové položky. Potomci elementu jsou tak přiřazeni této položce typu `CollegeOrUniversity`. Algoritmus končí s jednou zkonstruovanou položkou odpovídající příkladu z tabulky č. 1.

Omezení typů Ačkoliv atribut `itemtype` může pojmout několik hodnot oddělených mezerami, dle právě utvářených doporučení organizace W3C je zakázáno použití typů z různými jmennými prostory [12]. Toto chování není doporučováno, protože způsobuje víceznačnost při vyhodnocování relativních názvů vlastností v `itemprop` – algoritmus není schopný jednoznačně rozhodnout, ve kterému slovníku má takovou vlastnost přiřadit.

Srovnání s RDFa Technologie RDFa a HTML5 Microdata jsou si podobné zejména tím, že definují strukturovaná data v atributech elementů [12]. RDFa je stále aktivně podporované [11] a Microdata API pro Javascript nebylo nikdy prakticky implementováno do prohlížečů [1]. Přesto se Microdata dále vyvíjí a ačkoliv neposkytují kompletní možnosti technologie RDF, poskytují jednoduchý způsob notace strukturovaných dat. Ačkoliv Microdata nejsou schopna popsat jakoukoliv konstrukci RDF (nejedná se o vzájemně kompatibilní řešení), existují algoritmy pro konverzi Microdata do formátů skupiny RDF [12].

4.3.2 JSON-LD

JSON-LD přidává do formátu JSON možnost specifikace sémantických, propojených dat dle specifikace RDF. Na rozdíl od Microdat jsou u JSON-LD data v dokumentu úplně oddělená od obsahu HTML. Představuje přehledný a odlehčený způsob zadávání strukturovaných dat na web. V době psaní se také jedná o doporučovaný formát pro vyhledávač Google [10, 6].

Strukturovaná data jsou v JSON-LD formátována jako páry *atribut–hodnota* (či více hodnot jako pole) JSON objektu. Samotný objekt představuje popisovanou položku a atributy její vlastnosti. Názvy atributů začínající zavináčem `@` jsou rezervovány jako vyhrazené výrazy pro definici propojených dat. Mezi nejběžnější klíčová slova patří následující [10]:

- `@id` – Specifikace identity objektu, konkrétní instance. Očekává se IRI[8]. U knih se používá například ISBN, u jiných zdrojů se může uvést URL.
- `@type` – Určuje typ objektu (třídu). Lze uvést buď pojem s celým jmenným prostorem (absolutní URL), nebo je možné vytvoření aliasů pro daný kontext, jak popisuji v dalším bodě.
- `@context` – Slouží k uvedení jmenného prostoru (jinak řečeno kontextu). Můžeme tak přiřadit alias nějaké adrese URL a používat ji před pojmem s oddělením dvojtečkou (např. `schema:Person`) nebo uvést obecný kontext a zadávat přímo název pojmu (např. `Person`). Použijeme-li alias ze sekce `@context` přímo (bez dvojtečky), dosadí se jako URL pojmu.

[8]Identifikátor, rozšíření standardu URI o Unicode znaky [13]

Existuje ještě řada dalších klíčových slov a konstrukcí určených k zjednodušování, zkracování zápisu nebo pro definici sofistikovanějších struktur. Při správném použití aliasů (kontextu a jiných zkrácení) vzniká normovaná *kompaktní* forma. Pokud naopak všechny aliasy převedeme do plného tvaru, hodnoty na pole atd., získáme tzv. *rozšířený* tvar.

Do HTML dokumentu se JSON-LD přikládá značkou pro skript s upřesněním použitého typu. Mezi značkami již následuje samotný kód [2]:

```
<script type="application/ld+json">
{
   "@context": "http://schema.org",
   "@id": "http://lajcok.cz",
   "@type": "Person",
   "name": "Jiri Lajcok",
   "birthDate": "1997",
    "affiliation": {
        "@type": "CollegeOrUniversity",
        "url": "https://www.vsb.cz/",
        "name": "VSB"
    }
}
</script>
```

Výpis 3: Strukturovaná data ve formátu JSON-LD

Konzumenti strukturovaných dat na stránce hledají skripty typu JSON-LD, následuje interpretace JSON kódu uvnitř a zpracování propojených dat. Díky snadné notaci objektu v JSON je takový zápis jednoduchý a snadno zpracovatelný.

Kód 4.3.2 reprezentuje totožná strukturovaná data jako příklad v tabulce č. 1. Pro legální použití (bez penalt například v hodnocení vyhledávačů) musí tato odpovídat s obsahem viditelným uživateli. Skript by tedy měl být připojen k dokumentu podobnému kódu č. 4.3.1 [6].

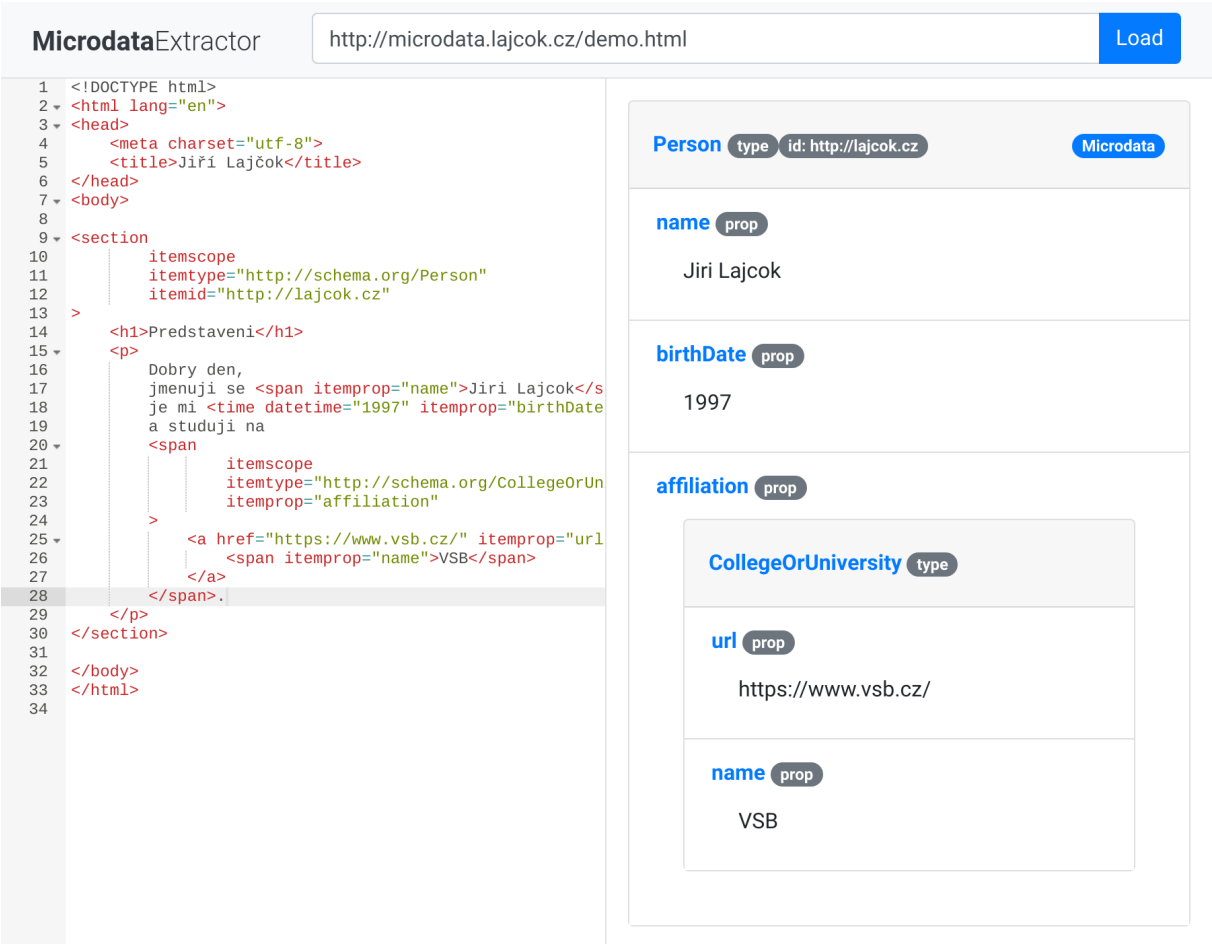
Nabízí se také možnost rozšíření stávajících API, která staví na technologii JSON, o propojená data. Ta pak může frontend[9] aplikace dynamicky připojit do dokumentu, toto řešení vyhledávací roboti dle dostupných zdrojů podporují [6].

[9] Část aplikace běžící v prohlížeči na klientské straně

5 Nástroj pro extrakci strukturovaných dat

V rámci této práce byla vyvinuta webová aplikace pro prohlížení strukturovaných dat z existujících internetových stránek. Poskytuje také dodatečné informace o správnosti integrace dat do dokumentu a kontrolu sémantiky podle `schema.org` slovníku.

Aplikace s pracovním názvem *MicrodataExtractor* je dostupná online na webové adrese `https://microdata.lajcok.cz`. Její zdrojové kódy jsou také součástí elektronické přílohy. V příloze A se pak nachází dokumentace nástroje.



Obrázek 4: Počáteční prostředí aplikace

5.1 Konkurenční nástroje

Nejedná se o unikátní nástroj a již existují podobná řešení jako například *Nástroj na testování strukturovaných dat* od společnosti Google. Implementací vlastní aplikace byl však umožněn náhled na vnitřní fungování extrakčních algoritmů a absolutní kontrola nad výsledky zpracování. Oproti nástroji od Google je *MicrodataExtractor* také striktnější v oblasti správnosti formátování, dodržování specifikace a korespondenci se slovníkem `schema.org`.

5.2 Technologie

Jedná se o webovou aplikaci psanou v programovacím jazyce TypeScript s využitím knihovny React pro tvorbu uživatelských rozhraní. Grafické prvky poskytuje knihovna Bootstrap. Pro správu Javascript balíků je použit nástroj NPM, který také zprostředkovává předdefinované akce pro běžné úkony jako spuštění vývojářského serveru, imitace backendu, sestavení projektu do balíčku nebo nasazení aplikace na sdílený webový hosting.

Většina logiky aplikace běží výhradně na klientské straně (tzv. *frontend*), ale pro získávání zdrojových kódů z jiných adres je nutné použití serverové části (tzv. *backend*), viz sekce č. 5.5. Serverová část je implementována jak v PHP (vhodné pro použití na sdíleném hostingu), tak i v Javascriptu pro NodeJS Express Server (určeno zejména pro vývoj na lokálním stroji).

TypeScript Jedná se o typový programovací jazyk, jehož kód je překládán do čistého Javascriptu. TypeScript (a obecně i Javascript či ECMAScript) implicitně využívá členění aplikace do modulů. Každý samostatný soubor zpravidla představuje modul a i celé adresáře mohou zaobalovat dílčí moduly a vystupovat jako celek.

React Knihovna React definuje tzv. komponenty určené ke konstrukci uživatelských rozhraní včetně jeho obsluhy v průběhu sezení. Vstupní atributy komponenty jsou označované jako `props`, kdežto proměnlivé stavy uchovává `state`. Volitelnou součástí knihovny React je také technologie JSX zjednodušující zápis při konstrukci DOM.

NPM Node Package Manager je balíkovací systém starající se o závislosti aplikace na externích balíčcích. Umožňuje instalaci součástí z dostupných zdrojů a správu jejich verzí včetně aktualizací.

Express Server Umožňuje vytváření backend serverů v jazyce Javascript. Lze v něm vytvářet například API pro frontend aplikaci.

PHP Pro použití na sdíleném webovém hostingu je backend část aplikace implementována i ve skriptovacím jazyce PHP.

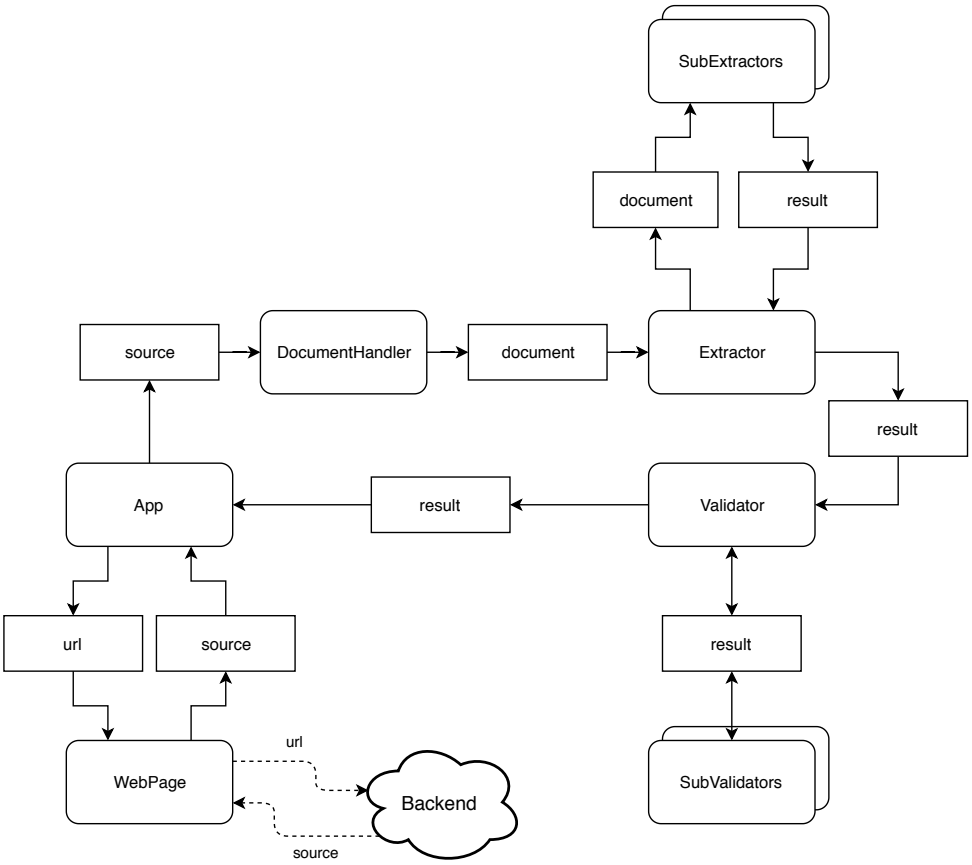
5.3 Struktura aplikace

Na vstupu nástroje stojí zdrojový kód dokumentu, očekávaným výstupem je pak výpis strukturovaných dat. Schéma (obrázek 5) zachycuje koloběh života aplikace. Počínaje blokem `App`, což je fakticky komponenta React a část uživatelského rozhraní. Jde o klíčový blok celého procesu a spouští se zde v závislosti na uživatelských akcích proces zpracování zdrojového kódu.

Vstupní uživatelská interakce může mít dvě podoby: načtení webového dokumentu z adresy URL či přímé zadání úryvku HTML kódu. V prvním případě dochází ke kontaktování serverové části s požadavkem na obsah dokumentu dostupného na požadované adrese – blok `WebPage`

zprostředkovávající komunikaci se serverovou částí. V druhém případě je zdrojový kód přímo zpracovatelný.

Komponenty vyžadují ke zpracování dokument, ne zdrojový kód. Blok `DocumentHandler` provádí tento převod. Následně je vyobrazena předávka dokumentu extrakčnímu bloku `Extractor`, která slouží jako obal na dílčí extrakční nástroje. Výsledky jsou pak spojeny a předány validátorům `Validator`, fungujícím na stejném modulárním principu jako extrakce. Zkontrolují výsledky, zapíšou logy a výsledek putuje zpět do `App` k zobrazení.



Obrázek 5: Zjednodušené schéma procesů a datových toků aplikace

5.4 Uživatelské rozhraní

Data jsou vyobrazena formou karet z knihovny Bootstrap. Obsahují typ nalezených položek, případně i jejich identifikátor. Zobrazuje se také zmínka o technologii, kterou byla daná položka získána. Následuje seznam jejich vlastností včetně hodnot. Vnořené položky jsou rekurzivně zobrazovány stejnou formou na místě hodnoty.

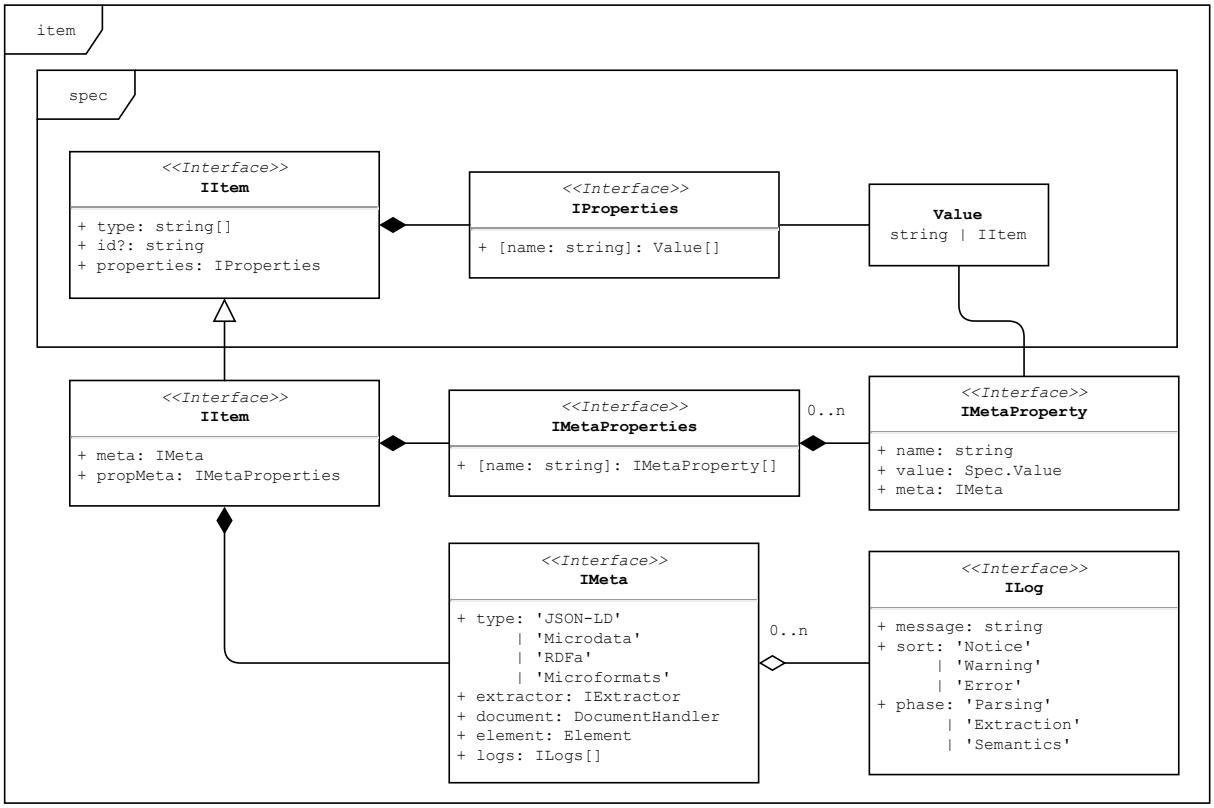
5.5 Backend

Jediným účelem serverové části aplikace je zprostředkování požadavků k získání zdrojového kódu z adresy URL. Toto řešení funguje jako proxy (čili prostředník) pro překonání bariéry způsobené tzv. *same-origin*[10] omezením.

Backend nabízí (nezávisle na použité technologii – PHP či Express Server) endpoint[11] dostupný v cestě `/curl.php`, kterému je přiřazen skript přijímající z dotazovacího parametru URL `?u=adresa` adresu k získání. Po obdržení zdrojového kódu je vrácen v těle odpovědi.

5.6 Datový model

Předmětem zpracování nástroje jsou tzv. *položky* (anglicky *item*). Aplikace definuje podobu objektové reprezentace dle datového modelu běžných formátů strukturovaných dat (viz sekce č. 4.1).



Obrázek 6: Datový model reprezentující položku v aplikaci

Základní podoba dle specifikace je vyjádřena rozhraním `Spec.IItem` – objekty tohoto typu jsou nosičem samotných dat. Pro analýzu položek je však zapotřebí rozšířeného modelu. Proto se v aplikaci prakticky vždy pracuje s typem rozhraní označovaného pouze jako `IItem`.

[10] Z bezpečnostních důvodů jsou prohlížečem zakázány klientské dotazy HTTP směřující na jinou doménu

[11] Endpoint, volně přeloženo *koncový bod*, je cesta, na které server odpovídá na klientské požadavky

5.7 Proces zpracování

V současné verzi se zpracování skládá ze dvou hlavních částí: extrakce a validace. Nástroje pro uvedené činnosti však nepracují přímo se zdrojovým kódem, nýbrž s objektovým modelem dokumentu (přístup DOM).

5.7.1 Extrakce

Extrakci zprostředkovává samostatný modul. Obsahuje dílčí moduly (každý pro jinou technologii) se společným rozhraním `IExtractor`, jejíž implementací lze přidat další extrakční třídy. Adresářový modul sjednocuje dílčí moduly do jednoho hlavního. Při extrakci se pak sekvenčně spouští jednotlivé dílčí extrakční třídy a jejich výsledky jsou nakonec sjednoceny. Extrakční nástroje vytváří nové položky.

5.7.2 Validace

Dále také existuje samostatný adresářový modul pro validaci. Validátor funguje analogicky jako v extrakční modulu – používají se dálčí validátory, které jsou spojeny do sekvenčních posloupností, které tvoří validační proces. Dílčí validátory implementují rozhraní `IValidator` a opět je možné rozšířit funkci modulu vlastní implementací. Na rozdíl od extrakce validátory pracují s existujícími položkami a pouze k nim přikládají validačními logy.

5.8 Implementované části

Vyvinutý nástroj poskytuje některá kompletní řešení pro účely zpracování případových studií, avšak některé nástroje nejsou implementovány kompletně nebo dle specifikací. Jsou poskytovány také prostředky pro rozšiřující implementace aplikace – nové extrakční třídy, validátory.

V současnosti aplikace podporuje extrakci strukturovaných dat formátovaných jako Microdata a zjednodušené JSON-LD. Třída pro Microdata následuje specifikaci dle standardu HTML [9], která je posledním platným standardem v době psaní. Nástroj pro extrakci JSON-LD podporuje zjednodušenou formu a pouze některé možnosti tohoto formátu. Implementace vychází z podoby používané v manuálech ke strukturovaným datům od Google [6].

Implementován je také právě jeden validátor pro slovník `schema.org` využívající oficiální balík definic [4].

6 Případové studie

Dle zadání byla vypracována studie o stavu sémantického webu v praxi. Jejím cílem je prozkoumat, kdo (tedy které subjekty – weby) a v jaké míře používá na webu strukturovaná data. K detekci sémantického obsahu na stránkách webu je použit nástroj *MicrodataExtractor* (viz sekce č. 5).

6.1 Pozorované parametry

Kromě náhodných postřehů budou pozorována určitá kritéria:

1. jaké sémantické informace web poskytuje a zda vůbec
2. použitý formát, slovník
3. kvalitativní parametry jako validnost, dodržování zásad a doporučení

6.2 Zastoupení subjektů

Přehled je zaměřen na některé kategorie webů a pro každou bude vybráno několik zástupců za cílem rovnoměrného zastoupení:

1. e-shopy
2. databáze a portály
3. zpravodaj a blogy
4. online video

6.3 Studie subjektů

Kompletní exportovaná data ze sběru jsou dostupná v elektronické příloze ve formátu `.csv`, z nichž také vychází jak text k subjektům, tak i statistické hledisko.

6.3.1 Alza.cz

Alza.cz je primárně český e-shop s elektronikou, ale nabízí také další typy zboží pro hobby, sport, drogerii nebo také auto-moto.

Na domovské stránce e-shopu byla zjištěna jediná položka – `BreadcrumbList` (popsáno dále). Dále na stránkách s produkty jsou zaznačeny jako `Product`. Obsahují typicky název, obrázek, popis, cenu zadanou ve vlastnosti `offer` a také identifikátory v rámci obchodníka (vlastnost `sku` – *Stock Keeping Unit*) či globální označení dílu výrobcem (`mpn` – *Manufacturer Part Number*). Nechybí ani informace o uživatelských hodnoceních definovaných hodnotou vlastnosti `aggregateRating`.

Na stránkách produktu Alza také označuje `BreadcrumbList` specifikující zařazení produktu do kategorií (podrobná navigace). V této položce dochází k zajímavému jevu – názvy jednotlivých položek `ListItem` obsahují HTML entity místo některých znaků (lze dobře pozorovat zejména u diakritiky). Entity ve značkách `<script>` podle definice HTML nejsou vyhodnocovány [14]. To má za následek zdeformované hodnoty jako např. `Série` (místo zjevného `Série`). V tomto případě záleží i na konzumentu obsahu – jak s takovou hodnotou naloží a také vůbec na způsobu interpretace. Interpret dle standardu XHTML například entity vyhodnotí i uvnitř značky `<script>`. *Nástroj na testování strukturovaných dat* od Google tyto hodnoty zobrazuje ve správné podobě – s vyhodnocenými entitami.

Kromě produktů jsou k dispozici také vyznačení článků (`NewsArticle`) či hodnocení (`Review`) formou článku. Značení těchto typů je však poměrně sporadické a nekonzistentní.

6.3.2 CZC.cz

Na úvodní stránce je popsán typ `WebSite` se specifikací vlastnosti `potentialAction`, které je přiřazena instance `SearchAction` – akce vyhledávání. Kromě adresy vyhledávací stránky (vlastnosti `target`) je specifikováno i `query-input`, což není pojem vyskytující se v `schema.org`. Tato konstrukce slouží pro zobrazení vyhledávacího pole u výsledků ve vyhledávači Google [7]. Jedná se tedy o účelové řešení specifického případu i přes nevalidní formuli, kterou však mohou ostatní konzumenti jednoduše ignorovat. Alternativně by Google mohl definovat pojem `query-input` ve vlastním a odkazovat se na tuto definici, což by bylo validním řešením, avšak to by mělo za následek zkomplikování kódu pro integraci funkce.

Každá zanořená stránka obsahuje také seznam `BreadcrumbList` označující zanoření v hierarchii stránek. Jejich implementace je však realizována Microdaty chybně a položkám seznamu `ListItem` na jejich elementu chybí atribut `itemprop="itemListElement"`, což má za příčinu, že se jeví jako položky nejvyšší úrovně a nepatří do `BreadcrumbList`.

Stránky produktů jsou vyznačeny jako `Product` včetně vlastností `offers` s cenou produktu, hodnocení `aggregateRating` a kódů produktů `sku` či `mpn`. Na webu CZC.cz se nacházejí také články, avšak jejich sémantické vyznačení chybí.

6.3.3 MALL.CZ

Každá stránka na MALL.CZ obsahuje `WebSite` popisující samotný web. Google nástroj rozpoznává navíc položku `BreadcrumbList` (rovněž na každé stránce). Její absence ve vyvinutém nástroji je způsobena striktně specifikačním chováním při extrakci Microdat, podle kterého je element položkou nejvyšší úrovně právě tehdy, když obsahuje atribut `itemscope` a zároveň neobsahuje `itemprop` [1]. Právě toto nastává v případě položky `BreadcrumbList` na e-shopu MALL.CZ.

Produkty jsou zaznačeny klasicky jako `Product` se specifikací nabídky `offers` a hodnocením `aggregateRating`. Hodnotou nabídky je položka `Offer` udávající cenu produktu, jejíž

měna je v dokumentu specifikována jako Microdata značkou `<span itemprop="priceCurrency" content="CZK"></span>`. Zaměřme se na vyznačení hodnoty vlastnosti `priceCurrency` – atributem `content`. Dle specifikace HTML5 je atribut `content` validní pouze na značce `<meta>` a proto ani vlastní vyvinutý nástroj tuto hodnotu neregistruje, místo toho použije obsahu tagu `<span>`, tedy prázdný řetězec [9]. V porovnání s tím nástroj od Google vyznačení interpretuje jako `CZK` z atributu `content`. Podle nově vyvíjené specifikace HTML5 Microdata[12] totiž umožňuje přepsání výchozího chování při zjištění hodnoty pro konkrétní tag právě atributem `content` (čímž povoluje také jeho použití na jakékoliv značce) [12].

Na rozdíl od některých konkurentů také například nespecifikuje vlastnost produktu `sku` (kód zboží specifický pro obchodníka).

6.3.4 Best Buy

Best Buy je zahraniční obchod (a také e-shop) s elektronikou působící především v USA. Pro případové studie na tomto subjektu bylo využito přímého vložení zdrojových kódů stránek z důvodů technické limitace vlastního nástroje, kdy nebylo možné získat zdroj tohoto webu.

Domovská stránka e-shopu obsahuje položku `WebSite` a také označení `Organization` popisující společnost. Kromě jejího názvu `name` jsou k dispozici adresy sociálních sítí pomocí vlastnosti `sameAs` a kontaktní místa `ContactPoint` pro zákaznickou podporu. Každá stránka má pak označení `BreadcrumbList` jako indikátor v hierarchii stránek. Jednotlivé položky `ListItem` pak obsahují explicitní specifikaci pořadí pomocí vlastnosti `position`.

Stránka detailu produktu obsahuje položku typu `Product` s typickými vlastnostmi jako `offers`, `sku`, `gtin13`. V případě vlastnosti `description` se v hodnotě vyskytují HTML značky, přičemž je využita technologie JSON-LD. Nejedná se o chybu jako takovou, ale pro konzumenty tento formát nemusí být očekávaný a zpravidla se očekává čistý text [7]. Strukturovaná data produktu obsahují chybu u vlastnosti `brand`, kterému je jako hodnota přiřazena položka typu `Thing`, přičemž je dle `schema.org` očekáván buď typ `Brand`, nebo `Organization`.

Best Buy u produktů poskytuje i položky typu `Review` s uživatelskými recenzemi v plném znění ve vlastnosti `reviewBody`. Kromě toho jsou přiložena data o obrázcích na stránce položkami typu `ImageObject` specifikující název `name` a adresu náhledu `thumbnailUrl`. Pro video je použit typ `MediaObject`, který kromě názvu a náhledového obrázku poskytuje navíc jeho délku `duration`.

Při návštěvě stránky s informacemi o kamenných prodejnách jsou poskytnuta data typu `ElectronicsStore` specifikující například lokalitu vlastností `geo` (hodnotou je instance třídy `GeoCoordinates`), adresu `address` (položka `PostalAddress`) či otevírací dobu `openingHours` (obsahuje množinu řetězců `Text`, kde jeden představuje hodiny v daný den).

[12] V době psaní práce je specifikace ve fázi *Working Draft* (pracovní koncept). Je kompletní a připravená stát se *Candidate Recommendation* (kandidátem k doporučení) [12]

6.3.5 ČSFD

ČSFD, tedy Československá filmová databáze, shromažďuje informace o filmech a seriálech. Mimo to poskytuje také přehledy o osobách ve filmu, hodnoceních či žebříčky.

Při načítání jakékoliv stránky z adresy URL se projevila chyba v kódování a získaný zdroj byl nepoužitelný. Pro správně formátovaný vstup bylo použito přímé vložení zdrojového kódu.

Samotné filmy (seriály či televizní programy) jsou jedinou entitou ve strukturovaném formátu. Stránky s detaily snímku obsahují jedinou položku `Movie` obsahující například vlastnosti s názvem `name`, obrázkem `image` a rok vytvoření v poli `dateCreated`. Mimo to jsou obsaženy také: jméno režiséra (vlastnost `director`) a sumarizované hodnocení (`aggregateRating`). Ačkoliv jsou poskytovaná strukturovaná data poměrně stručná, nebyly v nich zjištěny žádné chyby.

6.3.6 IMDb

IMDb zkracuje název International Movie Database a jedná se o obdobu českého ČSFD. Hned domovská adresa `www.imdb.com` obsahuje strukturované informace o organizaci (`Organization`) s řadou hodnot vlastnosti `sameAs` obsahujícími adresy URL profilů na sociálních sítích – tato konstrukce je rozpoznávána například znalostní bází Google, která tyto informace reprezentuje jako přehled sociálních sítí organizace [7].

Stránka s detailem snímku obsahuje entitu `Movie`, která kromě vlastností jako `name` či `image` definuje také žánry (`genre`), přístupnost snímku věkovým kategoriím `contentRating` a podrobně herce, režiséry a tvůrce jako položky typu `Person` (popřípadě `Organization`). Jsou přítomny také recenze – jak v sumarizované podobě (`aggregateRating`) tak i výběr jedné konkrétní (vlastnost `review` s hodnotou třídy `Review`) včetně uvedení autora (`author`) a obsahu (`reviewBody`). Nakonec jsou specifikovány také délka snímku (`duration`) a upoutávka (neboli `trailer`), kde hodnotu reprezentuje položka `VideoObject`.

IMDb poskytuje také profily osob a organizací spojených s filmem na samostatných stránkách, které rovněž obsahují strukturovaná data – položky `Person` či `Organization`. Tyto stránky jsou připojeny k datům o snímku (popisovaném v předchozím odstavci) pomocí vlastnosti `url` u položek označujících herce, režiséry atp.

6.3.7 Wikipedie

Wikipedie (anglicky označovaná jako Wikipedia), co by on-line komunitní encyklopedie, je v oblasti strukturovaných dat poněkud nekonzistentní – některé články označeny jsou, jiné nikterak.

V případě označených článků je využita technologie JSON-LD popisující položku typu `Article` s metadaty o názvu (`name`), datu publikování a změn (`datePublished`, `dateModified`), nadpisu (`headline`). Vlastnosti `sameAs` a `mainEntity` obsahují adresu směřující na entitu popisovanou v článku na stránce `www.wikidata.org`, kde je reprezentována heslovitými pojmy (avšak bez využití strukturovaných dat). Nakonec jsou součástí také údaje o autoru (`author`) a vydavateli (`publisher`).

6.3.8 Novinky.cz

Zpravodajský server Novinky.cz neobsahuje žádná strukturovaná data u článků ani na hlavní stránce.

6.3.9 Aktuálně.cz

Na úvodní straně webu Aktuálně.cz se vyskytuje popis položky typu `WebSite` včetně jejího vydavatele (vlastnost `publisher`).

Další strukturovaná data se nacházejí na stránce článku. Zde je přítomen článek (položka `NewsArticle`), včetně jejího vydavatele (jako tomu bylo na domovské stránce). Jednou z vlastností položky je `mainEntityOfPage` (*inverzní* vlastnost k `mainEntity`), jejíž hodnotou je webová stránka (položka třídy `WebPage`), pro níž je definován identifikátor obsahující adresu aktuální stránky. Opačný (inverzní) případ stejného významu by představovala konstrukce, kdy by hlavní položkou byl typ `WebPage` a ve vlastnosti `mainEntity` by byla specifikována hodnota typu `Article`. Nechybí ani specifikace klíčových slov – vlastnost `keywords`.

Na typu `Article` se zde vyskytuje také neexistující vlastnost `wordcount`, jež je pravděpodobně překlep skutečného názvu vlastnosti `wordCount` (počet slov článku). Vlastní nástroj hlásí v tomto případě správně chybu, protože identifikátory vlastností rozlišují velikost písmen [9]. Přesto však například nástroj pro testování strukturovaných dat od Google tuto vlastnost automaticky vyhodnotí a převede na `wordCount`.

6.3.10 CNN

Domovská stránka amerického zpravodajského serveru neobsahuje žádná strukturovaná data. Na stránce článku je však vyznačena položka typu `NewsArticle`, přičemž je však použita forma jmenného prostoru s protokolem HTTPS, absolutně tedy `https://schema.org/NewsArticle`. Ačkoliv se nejedná o chybu jako takovou, `schema.org` ve svých příkladech a specifikacích používá zpravidla tvaru s HTTP. To je také důvodem, proč vlastní extrakční nástroj ignoruje tento jmenný prostor při validaci. Tato forma byla dále pozorována u položek typu `VideoObject` nacházející se na totožné stránce.

Další položkou v dokumentu článku je `WebPage` popisující webovou stránku jako takovou (nyní již s protokolem HTTP). Zvláštností je v tomto případě vlastnost `speakable` sloužící ke specifikaci sekcí dokumentu, které jsou „vyslovitelné“ v podobě vhodné pro hlasové čtečky textu [4]. V případě této stránky je použita jako hodnota položka typu `SpeakableSpecification` s vlastností `cssSelector` – selektor CSS identifikující požadované elementy.

Aplikace na stránkách CNN hlásila špatně formátovaná data JSON-LD. Takové výskyty skriptů byly přeskočeny. Totožnou chybu hlásil i nástroj od Google.

6.3.11 BBC

Na úvodu webu britské společnosti BBC se vyskytuje položka `WebPage`. Kromě specifikace základních metadat stránky se zde nachází také vlastnost `mainEntityOfPage` s hodnotou udávající adresu URL aktuální stránky. Jako hodnota vlastnosti `publisher` (vydavatel) je použita položka typu `NewsMediaOrganization` (konkretizace obecné organizace `Organization` na publicistickou organizaci). Vlastní extrakční nástroj ji hlásí jako neexistující, což je však způsobeno faktem, že je tento typ v době tvorby práce ve fázi návrhu na plnou integraci a není tak zatím součástí aktuální implementace [4]. Tato položka navíc specifikuje vlastnost `publishingPrinciples` – tedy odkaz na principy vydávání článků.

Stránka článku obsahuje další typ schválený k budoucí integraci – `ReportageNewsArticle` (potomek `NewsArticle`) – konkretizující obecný novinový článek na reportáž. Přiřazené vlastnosti `publisher` a `author` využívají dříve popisovaného typu `NewsMediaOrganization`.

Seznam videí (typem `VideoObject`) přiložených ke článku je zadán vlastností `video`. Tento seznam využívá konstrukce `@list` v JSON-LD, která však není vlastním nástrojem podporovaná, což má za následek nesprávné zobrazení hodnoty vlastnosti.

V kategoriích článků se vyskytují kromě položek `WebPage` také strukturované seznamy podkategorií implementované položkou typu `ItemList` s jednotlivými položkami seznamu (`ListItem`).

6.3.12 YouTube

Domovská stránka serveru YouTube neobsahuje strukturovaná data. V dokumentu detailu videa se pak nachází hlavní položka `VideoObject`, jež obsahuje například vlastnosti `isFamilyFriendly` (je-li obsah přístupný všem věkovým kategoriím) či `genre` (žánr videa). Mimo tyto a další základní vlastnosti se zde vyskytují pojmy neexistující v `schema.org`: `paid`, `channelId`, `videoId`, `unlisted`. Dále se zde vyskytuje nesprávně zadaný pojem `embedURL`, formálně správný zápis je `embedUrl` (adresa videa ke vložení). Nakonec je zde také `interactionCount` (počet interakcí), jež je však již doporučen k nahrazení obecnější vlastností `interactionStatistic` (statistiky interakcí).

Nakonec jsou některé stránky vybaveny také navigací realizovanou položkou `BreadcrumbList`.

6.3.13 Vimeo

Úvod zahraničního webu Vimeo je osazen položku typu `WebSite` definující profily na jiných sítích (vlastnost `sameAs`) a potenciální aktivity na webu: `ViewAction` (sledování vizuálního obsahu), `SearchAction` (vyhledávání s využitím `query-input`, jak bylo popsáno např. v sekci 6.3.2).

Stránky s videi obsahují položku typu `VideoObject` detailně specifikující obsažené video. Součástí jsou mimo jiné tyto vlastnosti:

- `playerType` – typ přehrávače, např. technologie HTML5, Flash či jiné
- `embedUrl` – adresa pro vložení videa (správně zadaná oproti Youtube, viz 6.3.12)

- `videoQuality` – kvalita videa vyjádřena textem
- `thumbnail`, `thumbnailUrl` – vlastnosti specifikující náhledový obrázek videa (v případě Vimeo dvoumi různými způsoby)
- `interactionStatistics` – podrobně vyjádřené statistiky položkami `InteractionCounter` s vlastností `interactionType` (typ interakce) specifikována náležitou aktivitou z typů např. `WatchAction`, `LikeAction` či `CommentAction` (zadáno jako hodnota typu URL).

Nakonec nechybí stránky ani označení položkami typu `BreadcrumbList`.

6.3.14 Stream.cz

Český streamovací server Stream.cz neposkytuje žádná strukturovaná data.

6.4 Statistiky

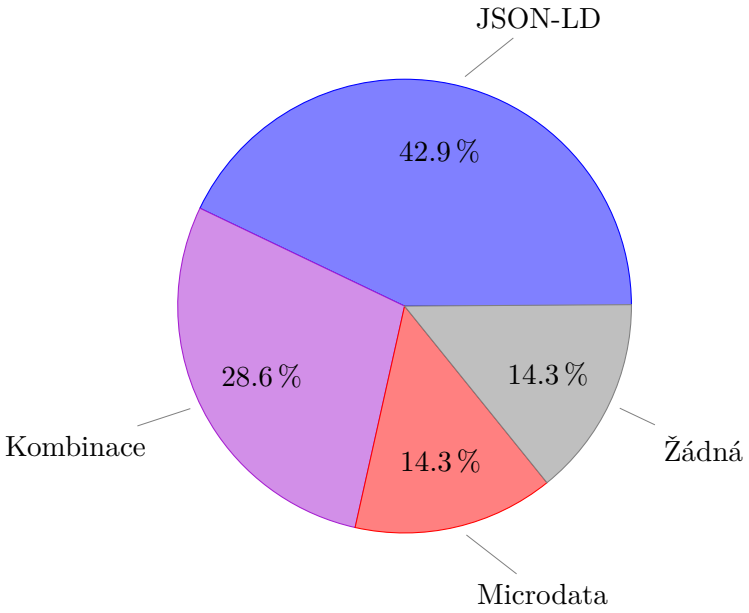
Ze zkoumaných 14 subjektů má 12 integrováno strukturovaná data ve svém webu. Většina (10 webů) využívá formát JSON-LD, Microdata se objevují na necelé polovině (6 webů). Kombinace obou předešlých nastává u 4 subjektů. Nakonec dva weby neposkytují žádná strukturovaná data. Pro údaje v grafické podobě viz obrázek č. 7. Účely použití technologií vyobrazuje tabulka č. 2, *stránka* označuje využití pro metadata dokumentu jako jsou `WebPage`, `WebSite` či `BreadcrumbList`.

Subjekt	**JSON-LD**	**Microdata**
Alza.cz	stránka, produkty, články	—
CZC.cz	produkty	stránka
MALL.cz	—	stránka, produkty
Best Buy	stránka, produkty	prodejny
ČSFD	—	filmy
IMDb	stránka, filmy, osoby	—
Wikipedie	články	—
Novinky.cz	—	—
Aktuálně.cz	stránka, články	—
CNN	stránka	články
BBC	stránka, články	—
YouTube	stránka, video	—
Vimeo	stránka, video	—
Stream.cz	—	—

Tabulka 2: Subjekty, technologie a jejich využití

Obrázek č. 7 reprezentuje podíly technologií včetně vyznačení překryvů a subjektů bez strukturovaných dat.

Tabulka č. 3 zobrazuje statisticky nejpočetnější výskyty typů slovníku `schema.org` v rámci zkoumaných subjektů. Není rozlišeno, zda je položka nejvyšší úrovně či zanořená.



Obrázek 7: Podíly použitých technologií u subjektů

Typ	**Subjekty**
`ImageObject`	8
`ListItem`	6
`Person`	6
`BreadcrumbList`	5
`AggregateRating`	5
`Organization`	5
`WebSite`	5
`VideoObject`	5
`Product`	4
`Offer`	3
`NewsArticle`	3
`Review`	3
`Rating`	3
`WebPage`	3

Tabulka 3: Nejpoužívanější typy na stránkách subjektů

Jako nejčastější typ chyby se projevila definice neexistujících vlastností – ať už typografickou chybou nebo vytvářením úplně nových, často i záměrně.

7 Závěr

V současnosti na poli sémantického webu dominují dvě technologie: Microdata a JSON-LD. Ačkoliv Microdata neposkytují veškerou funkcionalitu dle RDF, představují pohodlný způsob přidání sémantiky do již existujících dokumentů HTML. Při správném používání HTML je totiž jejich integrace poměrně přímočará, stačí tedy pouze přidat několik atributů a tím je hotovo. Využití si lze představit například u statických webových prezentací, kde se manipuluje přímo s dokumentem. Na druhou stranu JSON-LD poskytuje snazší integraci v případě webových aplikací nebo pokud již využíváme formátu JSON v API. Pak postačí jednoduše přidat propojená data do odpovědí HTTP a frontend aplikace je může zpracovat a přiložit k dokumentu.

Vlastní implementovaná aplikace zvaná *MicrodataExtractor* umožňuje nahlédnout do sémantiky existujících webů a dokumentů. Kromě toho, že mě osobně posunula ve vývoji aplikací pro frontend a porozumění (byť samozřejmě částečné) sémantickému webu, domnívám se, že aplikace má své využití. V aktuálním stavu má svá úskalí, avšak implementací dodatečných modulů a postupným laděním může vzniknout praktický analytický nástroj.

Případovým studiím bylo kladeno za cíl prozkoumat úroveň integrace sémantiky do různých oblastní na internetu. Je zřejmá motivace serverů pro e-commerce integrovat strukturovaná data za cílem zvýšení návštěvnosti či konverze. Tomu napomáhá zejména vliv společnosti Google jako vyhledávač a obecně v poli technologickém. Nejen u obchodníků se tak projevuje pragmatický přístup ke strukturovaným datům. Naproti stojí komunita, která je nadšená filozofií otevřeného propojeného a strojově čitelného webu. Ten by však neexistoval nebo by nebyl tak rozšířený bez komerčního vlivu. Obchodníci by se sice mohli prudce snažit o implementaci nových technologií, avšak i komunita stojící za schema.org zvolila přístup „*raději nějaká data, než žádná*“ [4].

Ze studií také plyne, že zahraniční servery více dbají na sémantiku a obecně technologickou stránku (například srovnání ČSFD–IMDb, Aktuálně.cz–BBC). Zda za tím stojí propracovanější marketing nebo větší pozornost technologickým trendům, to už zůstává pouhou spekulací. Může se zdát i překvapivá míra vyznačení video obsahu.

Filozofie Webu 3.0 se zaměřuje na formální definici informací pro její autonomní zpracování. Je to poznat z přílivu podpůrných technologií, které jsou aktivně vyvíjené. Nezávisle na použité technologii lze pozorovat každým dnem pokroky ve strojovém učení, což se projevuje na rychlosti při získávání informací – od vyhledávání až po virtuální a hlasové asistenty. Pro znalostní báze těchto strojů jsou však podstatná právě data a ta může poskytovat právě sémantický web.

Je možné, že se vývoj umělé inteligence posune až na úroveň, kdy bude schopen interpretovat lidskou řeči i grafickou organizaci webu. To však nemusí mít za následek úpadek sémantického webu. Strukturovaná data představují totiž univerzální jazyk, jehož zpracování bude navíc vždy efektivnější.

Literatura

[1] *DIVE INTO HTML5* [online]. Pilgrim, © 2009-2011 [cit. 2019-04-29]. Dostupné z: https://diveintohtml5.info/

[2] SIKOS, Leslie F. *Mastering structured data on the semantic web: from HTML5 microdata to linked open data*. New York, NY: Apress, [2015]. ISBN 978-1484210505.

[3] *Semantic Web* [online]. World Wide Web Consortium, © 2015 [cit. 2019-04-29]. Dostupné z: https://www.w3.org/standards/semanticweb/

[4] *Schema.org* [online]. Schema.org Community Group, 2019 [cit. 2019-04-29]. Dostupné z: https://schema.org/

[5] *Search Engine Statistics 2018* [online]. Smart Insights, 2018 [cit. 2019-04-20]. Dostupné z: https://www.smartinsights.com/search-engine-marketing/search-engine-statistics/

[6] Understand how structured data works. *Google Developers* [online]. 2019 [cit. 2019-04-15]. Dostupné z: https://developers.google.com/search/docs/guides/intro-structured-data

[7] *Google Developers: Search* [online]. Mountain View, CA: Google, 2019 [cit. 2019-04-29]. Dostupné z: https://developers.google.com/search/docs/guides/search-gallery

[8] *Google Knowledge Graph Search API* [online]. Mountain View, CA: Google, 2015 [cit. 2019-04-29]. Dostupné z: https://developers.google.com/knowledge-graph/

[9] *HTML Standard: Microdata* [online]. WHATWG, © 2018 [cit. 2019-04-29]. Dostupné z: https://html.spec.whatwg.org/multipage/microdata.html

[10] *JSON-LD 1.0: A JSON-based Serialization for Linked Data* [online]. World Wide Web Consortium, © 2010-2014 [cit. 2019-04-29]. Dostupné z: https://www.w3.org/TR/json-ld/

[11] TENNISON, Jeni. Microdata and RDFa Living Together in Harmony. *Jeni's Musings Home Blog* [online]. Tennison, 2011, Aug 20, 2011 [cit. 2019-04-29]. Dostupné z: http://www.jenitennison.com/2011/08/20/microdata-and-rdfa-living-together-in-harmony.html

[12] NEVILE, Chaals McCathie a Dan BRICKLEY. HTML Microdata. *World Wide Web Consortium* [online]. W3C, 2018, 26 April 2018 [cit. 2019-04-21]. Dostupné z: https://www.w3.org/TR/2018/WD-microdata-20180426/

[13] SUIGNARD, Michel a M. DUERST. Internationalized Resource Identifiers (IRIs). *Internet Engineering Task Force* [online]. The Internet Society, © 2005 [cit. 2019-04-15]. Dostupné z: https://tools.ietf.org/html/rfc3987

[14] *HTML 4: Document Type Definition* [online]. World Wide Web Consortium, 2018/03/20 [cit. 2019-04-29]. Dostupné z: `https://www.w3.org/TR/html4/sgml/dtd.html`

A Dokumentace nástroje

A.1 Vnitřní struktura

Adresáře představují základní vnitřní členění projektu, které zároveň odpovídá jeho interní logice. Následující seznam ji v bodech popisuje a uvádí také některé důležité soubory (respektive moduly, třídy a rozhraní aplikace).

- `package.json` – Soubor obsahuje metadata o aplikaci samotné coby balíčku NPM. Je zde uveden název balíčku, jeho verze a především závislosti na externích balíčcích (např. `typescript`, `react`). Obsahuje také servisní skripty spustitelné pomocí `npm run skript`.
- `tsconfig.json` – Jedná se o konfiguraci kompilátoru jazyka TypeScript.
- `public/`, `php/` – Složky zahrnují soubory určené k přiložení do výsledného, distribučního balíku. Konkrétně jde o soubory `public/index.html` (základní HTML dokument pro SPA[13]) a skript `php/curl.php`.
- `scripts/` – Servisní skripty pro NodeJS v separátních souborech `.js`, které jsou použity například ve skriptech spouštěných z `package.json`.
  - `php-faker.js` – Imituje funkci skriptu `/curl.php` na lokálním vývojovém serveru Express pro NodeJS. Lze spustit také pomocí `npm run php`.
  - `build.js` – Sestavuje aplikaci do produkčního balíku (angl. bundle) do adresáře `build/`. Spuštění je možné též provést pomocí `npm run build`.
- `src/` – Obsahuje vlastní zdrojové kódy aplikace. Jsou použity přípony `.ts` (standardní TypeScript soubor) nebo `.tsx` (kombinace TypeScript a technologie JSX).
  - `index.tsx` – Hlavní počáteční bod aplikace, ve kterém dochází k připojení hlavní komponenty `App`.
  - `util/` – Obsahuje elementární rozšiřující jazykové nástroje sloužící ke zjednodušení vývoje.
  - `components/` – Složka zahrnuje komponenty React určené k tvorbě uživatelského rozhraní.
  - `class/` – Sdružuje třídy a logiku doménového modelu aplikace.
    - `item/` – Obsahuje doménový model deklarovaný rozhraním položky (`IItem`) a další pomocné nástroje pro metadata (`IMeta`). Podsložka `spec/` obsahuje definice spojené čistě se specifikací datového modelu strukturovaných dat. Celý adresář představuje celistvý modul definovaný souborem `index.ts`.

[13] Single Page App – aplikace se nachází pouze na jedné stránce a veškeré dění probíhá dynamicky ve frontendu.

* `extraction/` – Sdružuje nástroje a proces pro extrakci dat z dokumentů. `IExtractor` deklaruje základní rozhraní pro extrakční nástroje včetně podoby jejich výsledků – `IExtractionResult`.
* `validation/` – Skupina validátorů s rozhraním `IValidator`, které zkoumají výsledky extrakce a případné chyby zaznamenávají do logů. Jediným implementovaným validátorem je `SchemaValidator`. Validátor používá specifikaci `schema.org`, jež je obsluhovaná podřazeným modulem v `schema/`). Klade si za účel validaci tříd položek a kontroly jejich vlastností.
* `tools/` – Obslužné třídy nástrojů užitečné k získávání a zpracování dokumentů.

A.2 Uživatelské rozhraní

Celá aplikace operuje na jediné stránce (princip SPA). Přístupy do aplikace jsou nejprve směřovány do `public/index.html`, která obsahuje základní HTML dokument. V těle dokumentu se nachází pouze upozornění pro klienty bez podpory Javascriptu a kořenový element pro aplikaci `<div id="root"></div>`. Při sestavování aplikace je do dokumentu přiložen také distribuční balík (typicky nazvaný `bundle.js`).

Sestavení balíčku počíná hlavním zdrojovým souborem – `src/index.tsx`. Ten zapříčiní připojení hlavní komponenty `<App/>` (modul `components/App.tsx`) k elementu s `id="root"`.

A.2.1 Komponenta `<App/>`

Komponenta reprezentuje celé okno aplikace a spravuje dílčí komponenty. Uchovává aktuální stav zdrojového kódu v `state.source` a výsledků extrakce v `state.result`.

Podstatnou část obrazovky zabírají náhledy zdrojového kódu (vlevo, `<CodePreview/>`) a výsledky extrakce (vpravo, `<ItemsPreview/>`). Vrchní částí vede lišta s titulkem a vstupem pro zadání URL adresy (`<InputURL/>`). Design aplikace je responzivní, avšak nejpohodlnější použití nabízí horizontální zobrazení a ovládání myší na stolním počítači či laptopu.

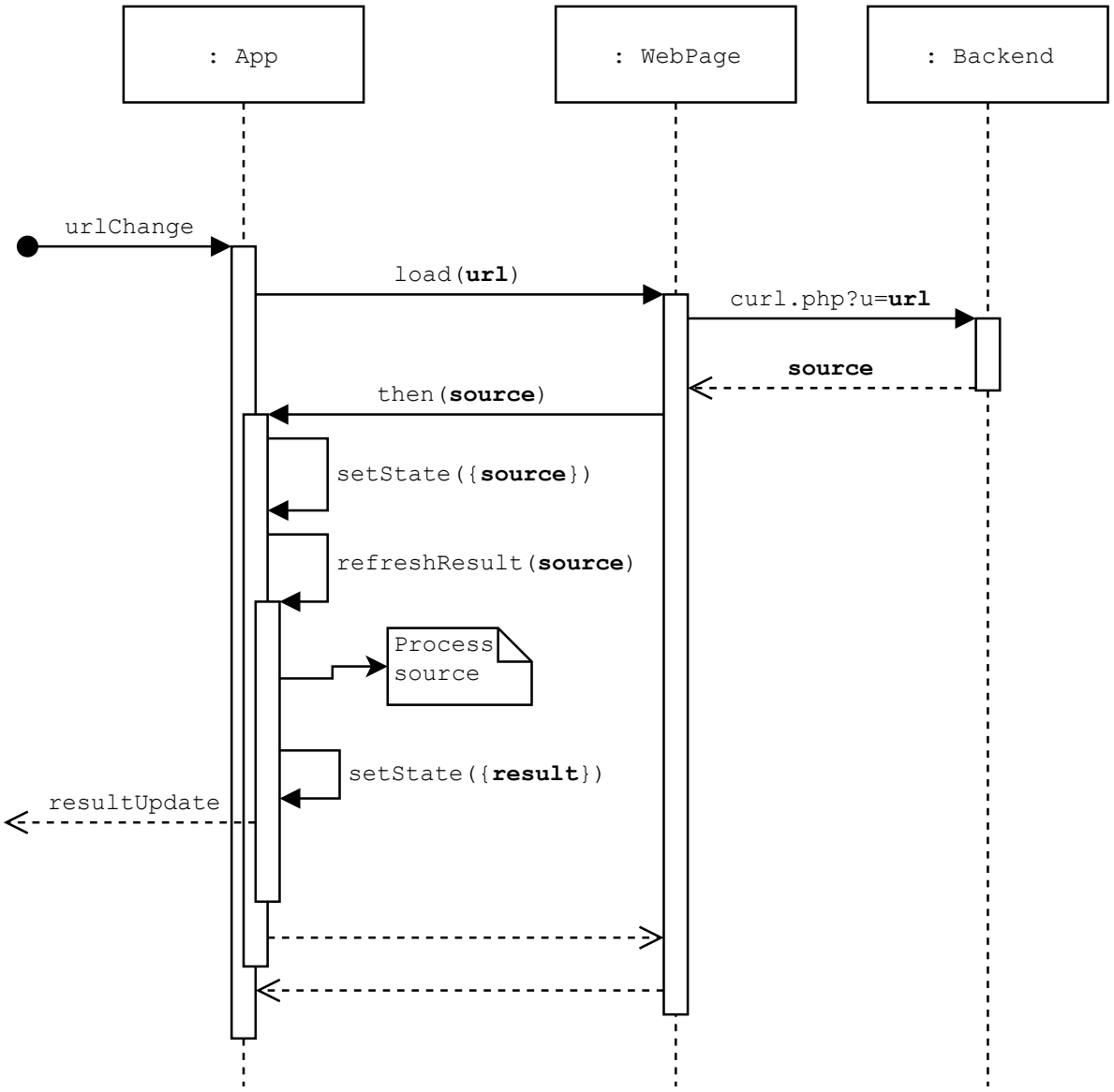
Mimo vizuální organizaci prostoru také obstarává uživatelské vstupy (změna v kódu či načtení z adresy URL) a v reakci na ně spouští proces extrakce a validace. Tyto procesy popisují diagramy v obrázcích 8 a 9.

A.2.2 Komponenta `<InputURL/>`

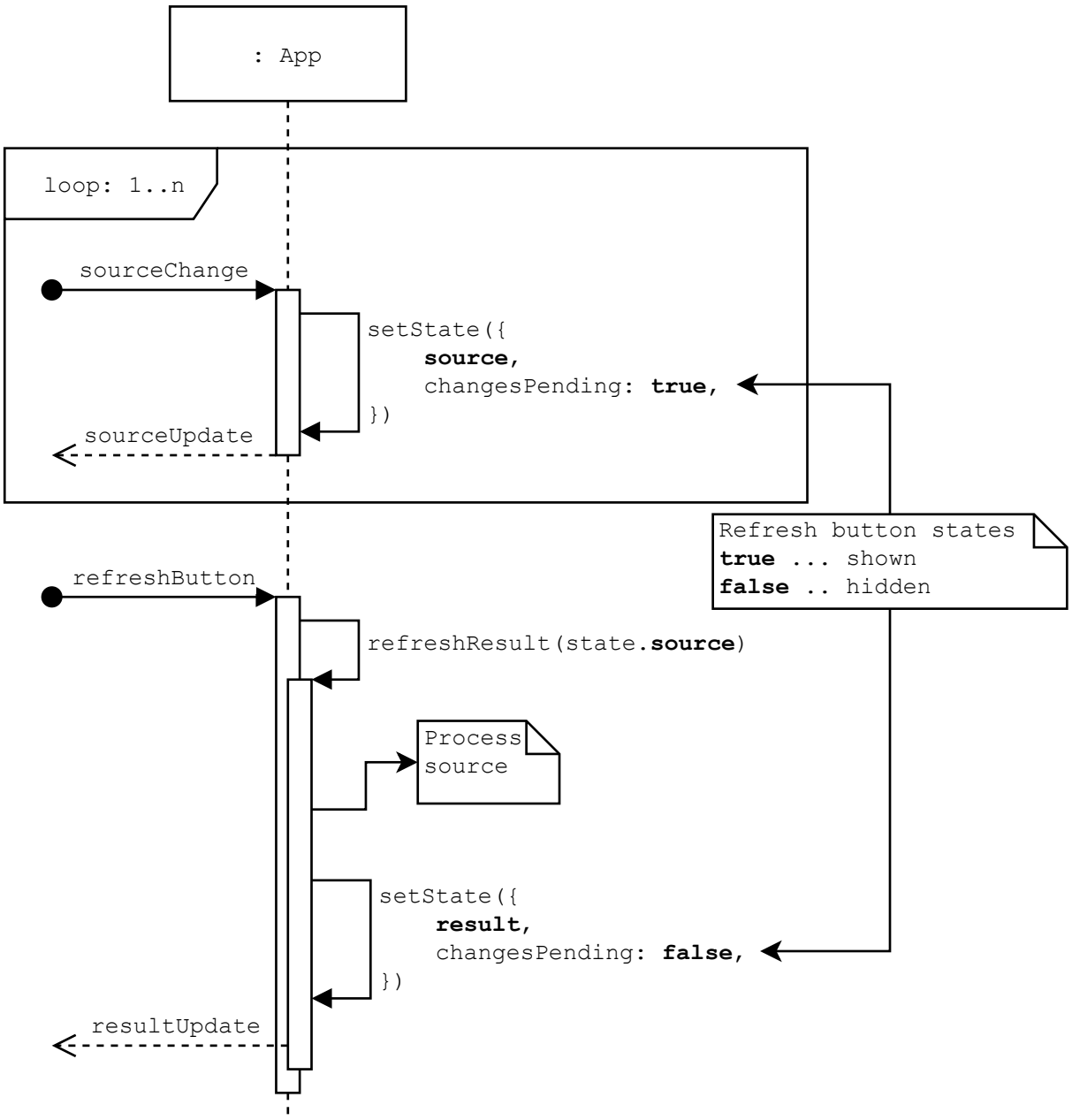
Jedná se o obal formuláře s polem pro zadávání adresy URL. Vstupními `props` lze jako atribut definovat výchozí adresu v poli (`props.url`) a předat callback[14] funkci (`props.onChange`) pro uvědomění o změně URL při odeslání formuláře.

Interně se pro formulářové pole `<input type="url"/>` využívá tzv. *kontrolovaného vstupu*. To znamená, že aktuální hodnota pole je vždy v synchronizaci s vnitřním stavem kompo-

[14]Funkce předávaná jako parametr funkci, metodě či konstruktoru, která se zavolá (angl. *invoke*) zevnitř.



Obrázek 8: Sekvenční diagram zpracování dokumentu z adresy URL



Obrázek 9: Sekvenční diagram zpracování ručně zadaného kódu

nenty (`state.url`). Při každé změně na vstupu se vždy `state.url` provede aktualizace stavu – `setState({url: url})`. Naopak je zase hodnota pole vždy určena aktuálním stavem: `value={state.url}`.

A.2.3 Komponenta `<CodePreview/>`

Slouží jako adaptér pro externí editor kódu AceEditor (z NPM balíčku `react-ace`) se stejnojmennou komponentou `<AceEditor/>`. Poskytuje pohodlné prohlížení i psaní kódu včetně skrývání bloků, zvýrazňování syntaxe a základní kontrolu jazyka.

Komponenta `<CodePreview/>` obaluje AceEditor a zprostředkovává komunikaci se zbytkem aplikace – zejména vzájemnou výměnu obsahu. Mimo to také umožňuje responzivní chování editoru.

A.2.4 Komponenta `<ResultPreview/>`

Tato komponenta je vizuální reprezentací výsledku extrakce v přehledné struktuře složené z elementů Bootstrap třídy `class="card"`. Umožňuje rekurzivní zobrazení vnořených položek včetně logů. Názvy pojmů jsou aktivními odkazy směřujícími na definici ze slovníku. Pro přehlednost jsou pojmy zkráceny na pouhý název v případě, že obsahují jmenný prostor `http://schema.org`. Vícero hodnot vlastností je seskupeno do jedné skupiny.

A.3 Datový model

Předmětem zpracování jsou položky (anglicky též *item*). Datový model domény se tedy zaměřuje téměř výhradně pouze na entitu reprezentující položku.

A.3.1 Rozhraní `Spec.IItem`

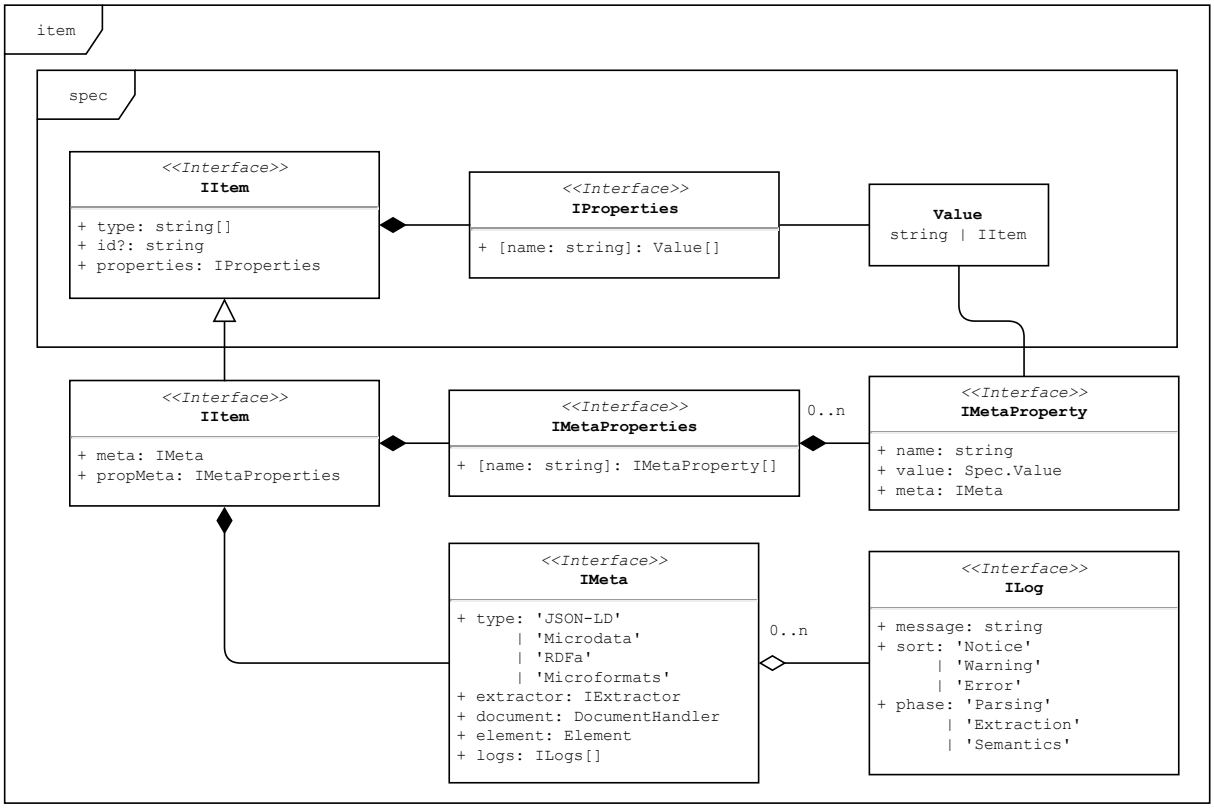
Reprezentuje položku tak, jak ji popisují specifikace. Obsahuje tak holé minimum pro držení kompletních informací položky extrahované z dokumentu a konstrukci hierarchické struktury s využitím rekurzivních referencí.

Rozhraní `Spec.IItem` z modulu `src/class/structured-data/item` je součástí pod-modulu `Spec`, který je pouze odkazem na modul nacházející se v podsložce `./spec`. Modul `Spec` obsahuje také pod-modul `util` s podpůrnými funkcemi a tzv. *type guard*[15].

`Spec.IItem` nabízí následující veřejné rozhraní atributů:

- `type: string[]` – typ (typy) položky uchované jako pole řetězců
- `id?: string` – nepovinný identifikátor v řetězci
- `properties: IProperties` – struktura uchovávající vlastnosti položky, jejíž definice je pro snadné rozšiřování oddělena

[15]Funkce type guard v jazyce TypeScript ověřuje, zda je výraz požadovaného datového typu a následně toto zjištění používá k dalším typovým kontrolám či nápovědám.



Obrázek 10: Datový model reprezentující položku v aplikaci

Rozhraní `IProperties` je definováno jako páry *název–pole hodnot* (`[name: string]: Value[]`). Pro hodnoty (`Value`) slouží opět externí definice coby typový alias pro řetězec nebo podřazenou položku: `string | Spec.IItem`. Pro zjištění typu hodnoty existují v modulu `util` zmiňované type guard funkce `isString(hodnota)` a `isItem(hodnota)`.

A.3.2 Rozhraní `IItem`

Ačkoliv je název samotného rozhraní shodný jako u `Spec.IItem`[16], jsou rozlišena specifikací modulu, ze kterého pochází. `IItem` rozšiřuje `Spec.IItem` o dodatečné atributy sloužící k předávání širších informací o položce jako například metadata extrakce – dokument a element, ze kterého položka pochází, nebo logy s chybami, kterých se využívá rovněž při validaci.

Rozhraní přidává následující atributy:

- `meta: IMeta` – přídavná metadata položky formou definovanou dalším rozhraním
- `propMeta: IMetaProperties` – podobné `IProperties` s tím rozdílem, že místo přímého specifikování hodnoty jsou uchovávány v obalovacím rozhraní `IMetaProperty` následující struktury:
  - `name: string` – název vlastnosti jako řetězec
  - `value: Value` – hodnota s typem již definovaného aliasu `Value`
  - `meta: IMeta` – metadata vlastnosti

Rozhraní pro metadata `IMeta` je navrženo k nesení dodatečných informací položek a jejich vlastností. Mezi nejpodstatnější patří:

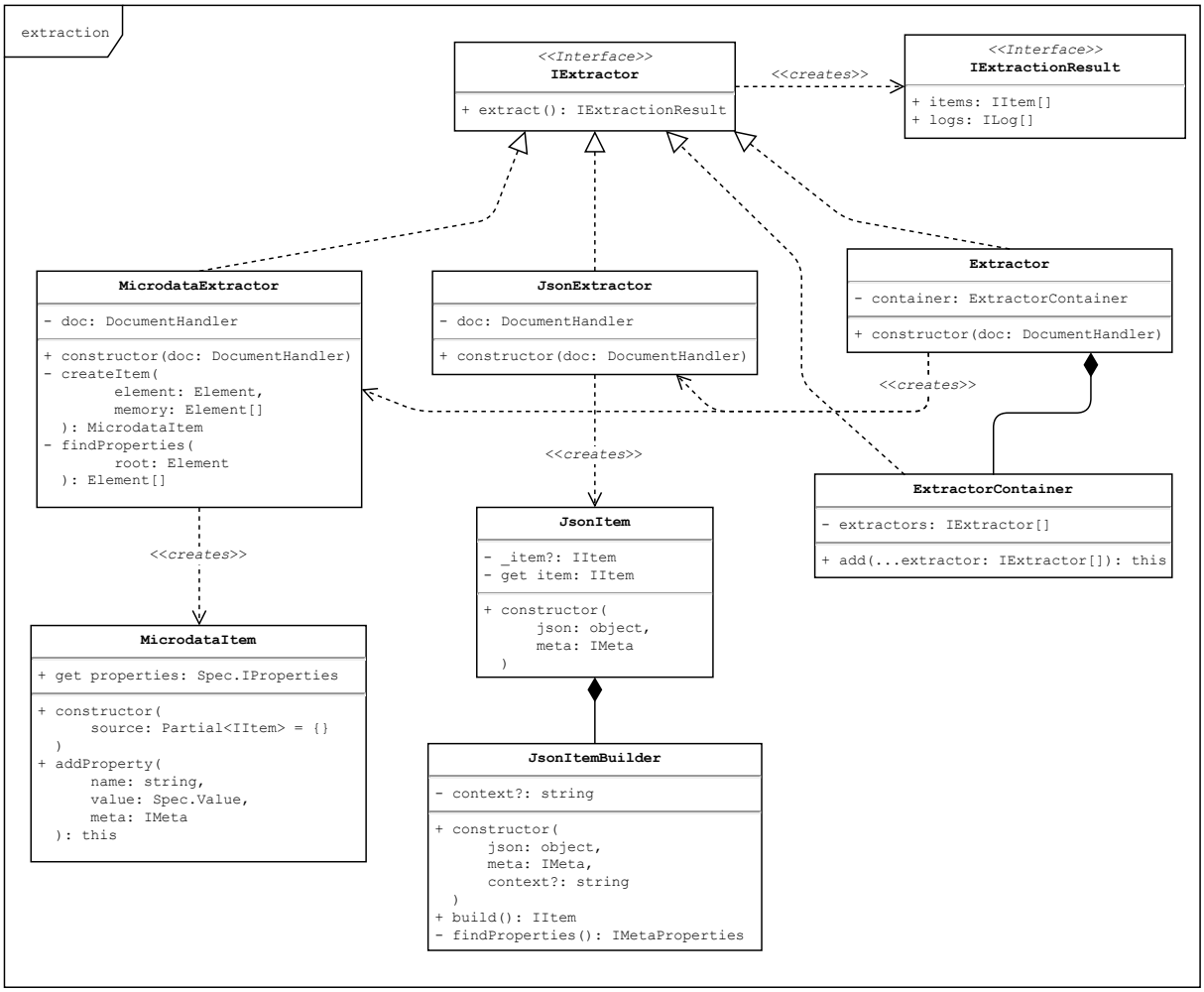
- `type: 'JSON-LD' | 'Microdata' | ...` – atribut určující název technologie, kterou byla data kódována, datového typu *string literal*[17].
- `logs: ILogs[]` seznam logů sbíraných v průběhu extrakce či validace reprezentovaný polem instancí `ILog` strukturovaných následovně:
  - `message: string` – textový obsah logu obsahující zdůvodnění, specifikaci chyby či možné řešení
  - `sort: 'Notice' | 'Warning' | 'Error'` – kategorizace logů do úrovní:
    - `'Notice'` – zjištění, která má cenu zmínit, ale nejedná se o chybu
    - `'Warning'` – varování na problém, který lze přeskočit či ignorovat
    - `'Error'` – závažná chyba, která znemožňuje zpracování dat
  - `phase: 'Parsing' | 'Extraction' | 'Semantics'` – fáze zjištění dané skutečnosti
    - `'Parsing'` – fáze rozboru, zpracování kódu (např. špatné formátování JSON)

[16] Dále v textu je striktně dodržován rozdíl mezi označeními `Spec.IItem` a `IItem`.
[17] Volně přeloženo jako *doslovný řetězec*. Typ, jehož hodnota musí nabývat právě jednoho řetězce z výčtu.

* `'Extraction'` – při získávání strukturovaných dat (např. cykly v grafu)
* `'Semantics'` – interpretace významu dat (např. hledání pojmu ve slovníku)

A.4 Proces extrakce

Modul `src/class/structured-data/extraction` obsahuje jak konkrétní implementace rozhraní `IExtractor`, tak i pomocné nástroje pro konstrukci extrakčního procesu. Nakonec nabízí také řešení připravené k okamžitému použití bez nutnosti zkoumat interní funkčnost modulu. Předmětem samotného rozboru je zpravidla objekt typu `DocumentFragment` (standardní třída Javascriptu pro manipulaci DOM) nesený instancí třídy `DocumentHandler` (z modulu `src/class/tools/DocumentHandler.ts`).



Obrázek 11: Třídní diagram modulu pro extrakci

A.4.1 Rozhraní `IExtractor`

Rozhraní sdružuje a definuje jednotnou formu pro extrakční nástroje. Deklaruje jediný prvek – metodu `extract(): IExtractionResult` pro vyvolání extrakce. Návratový typ je dále definován rozhraním `IExtractionResult` s atributy:

- `items: IItem[]` – nalezené položky uchované v poli
- `logs: ILog[]` – logy vztahující se k extrakci samotné

A.4.2 Třída `MicrodataExtractor`

Třída implementuje `IExtractor` a zabývá se extrakcí dat strukturovaných pomocí technologie HTML5 Microdata. Extrakční algoritmus následuje specifikaci HTML standardu pro maximální konformitu [9].

Modul třídy (`MicrodataExtractor.ts`) obsahuje další interní třídu `MicrodataItem` (implementace `IItem`) pro zjednodušení procesu sestavování položky. Řeší zejména přidávání a třídění vlastností do požadované struktury, o které se stará její metoda `addProperty(...)`. Mimo to používá pro atribut `properties` tzv. *property getter*, který její hodnotu na vyžádání aktivně vytvoří na základě dat z `propMeta` s cílem zajistit jednotný původ dat dle principu *single source of truth*.

Metoda `findItems()` Vyhledávají všechny elementy, které definují novou položku (nesou atribut `itemscope`) nejvyšší úrovně (není vlastností, tedy nemají atribut `itemprop`), dotázáním se na dokument:

```
doc.querySelectorAll('[itemscope]:not([itemprop])')
    .map(element => createItem(element));
```

Výpis 4: Microdata – získání položek nejvyšší úrovně

Metoda `createItem()` Nalezené elementy jsou předány metodě `createItem()` za cílem jejich převedení na objekt položky `IItem`, konkrétně `MicrodataItem`. Jako parametry přijímá `element` ke zpracování a nepovinně paměť `memory` (pole již zpracovaných elementů ve stromu).

```
private createItem(element, memory = []) {
    memory.push(element);
    const item = new MicrodataItem({
        type: splitTokens(element.getAttribute('itemtype')),
        id: element.getAttribute('itemid'),
    });

    for (const property of findProperties(element)) {
        let value;
        if (property.hasAttribute('itemscope')) {
            if (!memory.includes(property)) {
                // Rekurze pro podřazené položky
                value = createItem(property, memory.slice());
            } else {
                // Rekurze pro podřazené položky
                value = 'ERROR';
            }
        } else {
            // Zpracování hodnoty vlastnosti
            value = getValue(property);
        }

        const names = separateTokens(property.getAttribute('itemprop'))
            .map(token => combine(token, item.type));

        // Přidání zpracované vlastnosti do objektu
        item.addProperty(names, value);
    }

    return item;
}
```

Výpis 5: Microdata – algoritmus tvorby objektu položky

Metoda `findProperties()` Vyhledávání vlastností je poskytováno metodou `findProperties()`. Na vstupu stojí kořen vyhledávání – element položky.

```
private findProperties(root) {
   const results = [],     // Výsledky hledání
         memory = [root],  // Pamět již nalezených
         pending = [];     // Čekající na zpracování

   // Vyhodnocení ‘itemref‘ a nalezení jednotlivých elementů
   for(const id in separateTokens(element.getAttribute('itemref'))) {
      pending.push(doc.getElementById(id));
   }

   while(pending.length > 0) {
      // Bere element z fronty
      const current = pending.shift();

      // Přeskočení kroku cyklu, pokud byl již element zpracován
      if(!memory.includes(current)) {
          memory.push(current);
        } else {
         continue;
      }

        if (current.hasAttribute('itemscope')) {
         // Je-li nalezená položka je vlastností, přidá se do výsledků
            if (separateTokens(current.getAttribute('itemprop')).length > 0) {
                results.push(current);
            }
        } else {
         // V opačném případě se postupuje hlouběji
            pending.unshift(...current.children);
        }
   }

   return results;
}
```

Výpis 6: Microdata – algoritmus vyhledávání vlastností

A.4.3 Třída `JsonExtractor`

Ačkoliv existuje oficiální knihovna pro extrakci dat ve formátu JSON-LD, pro účely demonstrace technologie a práce s ní je použit vlastní, zjednodušený algoritmus. Nepodporuje všechny konstrukce formátu dle specifikací, ale je zaměřen na její nejpoužívanější formulace na webu.

Stejnojmenný modul se skládá celkem ze tří tříd, které jako celek zprostředkovávají extrakci. Hlavní třída modulu `JsonExtractor` implementující rozhraní `IExtractor` je jediným externě exportovaným prvkem. Samotný proces sestavení položky poskytuje třída `JsonItemBuilder` a výsledkem je instance `JsonItem`.

- `JsonExtractor` – Hlavní třída implementující rozhraní `IExtractor` je jediným exportovaným prvkem. Vyhledává v HTML dokumentu JSON-LD data ze značek `<script type="application/ld+json">`, zpracuje je jako čisté JSON objekty (pole objektů jsou rozdělena) a vytvoří z nich instance `JsonItem`.

- `JsonItemBuilder` – V konstruktoru třída přijímá objekt JSON a interpretuje je jako sémantická data. Zároveň implementuje `IItem`, takže definuje atributy rozhraní jako *property-getter* – data se extrahují z JSON dynamicky na vyžádání při každém přístupu. Je tedy zřejmé, že tento stav je pouze přechodný, proto třída deklaruje také metodu `build()`, která vrací objekt `IItem` s klasickými statickými atributy.

- `JsonItem` – Třída reprezentuje položku extrahovanou z JSON-LD. V konstruktoru přijímá JSON data, avšak třída samotná tato data neinterpretuje. Slouží pouze jako zprostředkovatel a předává je nově vytvořené instanci `JsonItemBuilder`. Atributy rozhraní `IItem` fungují jako *property-getter* – k extrakci dochází ve skutečnosti až na vyžádání vyvoláním `build()`, přičemž je výsledek uložen pro další přístupy (princip návrhového vzoru *lazy-load*).

Modul podporuje formu JSON-LD s použitím jednoduchého kontextu (hodnota `@context` je adresa jmenného prostoru v řetězci) nebo úplně bez něj. Stejně tak jsou i `@id` a `@type` správně zpracovány pouze v konkrétních případech popsaných dále. Naopak není podporováno definování aliasů, použití klíčových vlastností jako `@graph` nebo `@value`.

Třída `JsonItemBuilder` mapuje atributy z `IItem` na vstupní JSON objekt následovně:

- `type` – Typu odpovídá pole hodnot z `@type`. Jedná-li se o primitivní typ (řetězec), je nejprve převeden na jednoprvkové pole. Není-li `@type` definováno vůbec, zůstává pole prázdné. Všechny dílčí typy jsou zkombinovány s kontextem (je-li definován) pro získání absolutních identifikátorů.

- `id` – Identifikátor je mapován jedna ku jedné atributu `@id`, očekává se řetězec `string`, nebo `undefined`.

- `propMeta` – Každá vlastnost JSON objektu, jejíž název není klíčovým slovem (začínajícím `@`), představuje rovněž vlastnost nové položky. Názvy vlastností jsou při mapování opět

zkombinovány s kontextem. Není-li hodnota polem, je na pole převedena (vzniká jednoprvkové pole). Následně jsou dílčí hodnoty postupně zpracovávány a obalovány do struktury `IMetaProperty` (název vlastnosti, hodnota a metadata).

  - Pro hodnotu typu objekt je vytvořena nová instance `JsonItemBuilder` (rekurze), které je objekt předán v konstruktoru.
  - Ostatní typy hodnot (primitivní) jsou převedeny na řetězec.

- `properties` – Jedná se o *property-getter* mapovaný na atribut `propMeta`, který jej dynamicky převádí do požadované struktury (dle rozhraní `IProperties`).

A.4.4 Třída `ExtractorContainer`

Pro konstrukci procesů extrakce slouží třída `ExtractorContainer`. Jedná se o další implementaci `IExtractor`, která umožňuje spojení dílčích extrakčních nástrojů ve specifikovaném pořadí a jejich kombinaci používat jako jeden celek (princip návrhového vzoru `composite`). Dílčí výsledky se spojují do jednoho.

A.4.5 Třída `Extractor`

Třída je určena ke zjednodušení použití jednotlivých nástrojů a extrakčního modulu jako celku. Spojuje tak obě implementované třídy pro extrakci `MicrodataExtractor` a `JsonExtractor` do struktury `ExtractorContainer`. Nabízí tak hotové řešení připravené k okamžitému použití.

A.5 Proces validace

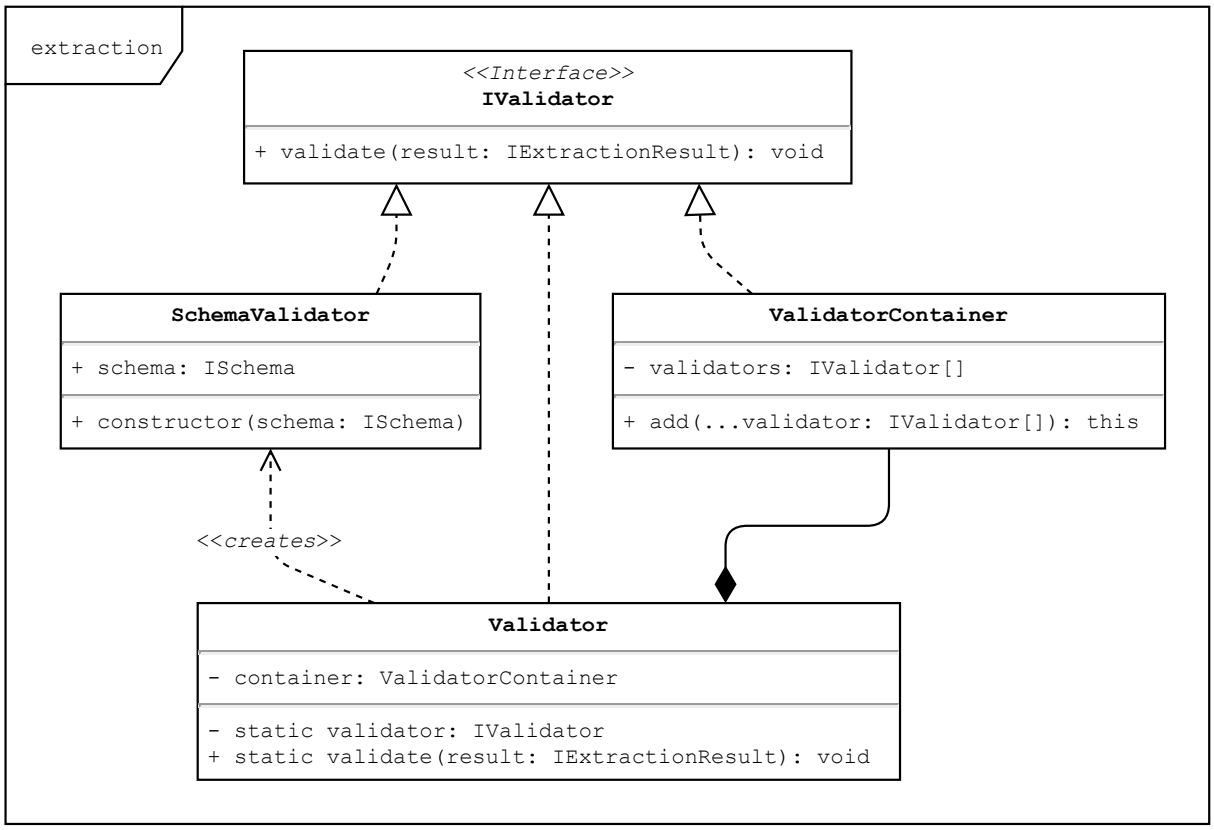
Validaci obecně zprostředkovává modul `src/class/structured-data/validation`. Jedná se o jakýkoliv typ kontroly již extrahovaných výsledků. Aplikace v současném stavu poskytuje pouze validaci podle definic slovníku `schema.org`, ale lze je snadno rozšířit implementací společného rozhraní pro validátory `IValidator`.

Podobně jako v případě extrakce modul nabízí jak obecné definice, tak i konkrétní implementace a kompletní řešení ve formě předdefinovaných nástrojů.

A.5.1 Modul `schema`

Obsahuje oficiální definice `schema.org` ve formátu JSON-LD v souboru `schema.json`[18]. Pro snadný přístup k datům je k dispozici třída `Schema`, která umožňuje získávání pojmů podle jejich identifikátoru (`get(id)`) či pojmy vlastností náležící k dané třídě (`getProperties(class)`). Pro optimalizaci také výsledky uchovává v mapě. Rozhraní `ITerm` (a potomci `IClass`, `IProperty`) představují definici pojmu používanou v aplikaci. Nakonec je poskytnuta také třída `TermLoader` sloužící jako reference na ještě nenačtené pojmy (princip fungování *lazy-load*).

[18] Přípona `.json` je použita kvůli implicitní podpoře jazyka TypeScript při importování



Obrázek 12: Třídní diagram modulu pro validaci

A.5.2 Rozhraní `IValidator`

Validace se vyvolá použitím jediné metody rozhraní: `validate(result: IExtractionResult): void`. Jako parametr přijímá výsledek extrakce `IExtractionResult` a není deklarována žádná návratová hodnota. Validátor totiž manipuluje přímo s logy výsledku, zejména přidává vlastní na základě jeho zjištění.

A.5.3 Třída `SchemaValidator`

Jedná se o konkrétní implementaci validačního rozhraní a poskytuje kontrolu výsledků extrakce proti definicím `schema.org`. Zpracovává pojmy výhradně ze jmenného prostoru `http://schema.org/`, ostatní jsou ignorovány. Následující parametry položek jsou ověřovány:

- existence třídy položky ve jmenném prostoru `schema.org`
- náležitost vlastnosti do domény třídy (existuje-li na některém z typů položky)
- splnění restrikce rozsahu typů hodnoty (respektive třídy položky[19]) přiřazené k vlastnosti

A.5.4 Třída `Validator`

Třída představuje hlavní validátor, tedy kompletní řešení zastupující celý modul. Interně využívá `ValidatorContainer` (analogie ke třídě `ExtractorContainer` popisované v sekci A.4.4), jehož náplní je jediný implementovaný validátor `SchemaValidator`. Opodstatněním této konstrukce je konformita s modulem `src/class/structured-data/extraction` a také možnost rozšíření dalšími validátory.

[19] V aktuální verzi aplikace není správně implementováno odvození tříd (potomků) podle stromu dědičnosti. Například pokud vlastnost očekává instanci třídy `Thing` (rodič všech tříd ve `schema.org`), bude nesprávně hlásit chybu, přiřadíme-li jí instanci typu `Book`. Striktně požaduje přesný typ `Thing`.